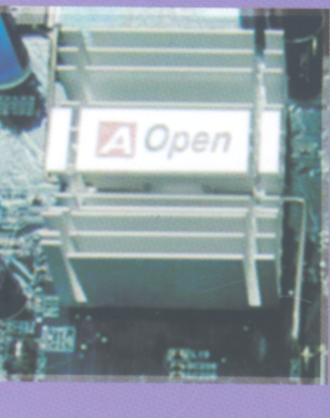
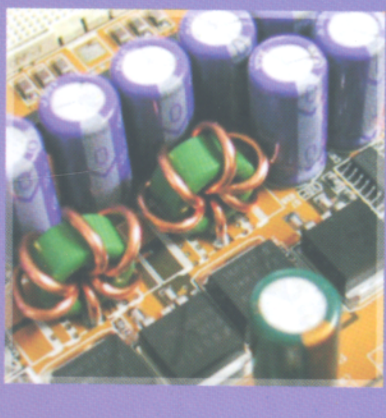
SỞ GIÁO DỤC VÀ ĐÀO TẠO HÀ NỘI

GIÁO TRÌNH

Cấu trúc máy tính

DÙNG TRONG CÁC TRƯỜNG TRUNG HỌC CHUYÊN NGHIỆP







SỞ GIÁO DỤC VÀ ĐÀO TẠO HÀ NỘI

TS. **PHÓ ĐỨC TOÀN**

GIÁO TRÌNH
CẤU TRÚC MÁY TÍNH

(Dùng trong các trường THCN)

NHÀ XUẤT BẢN HÀ NỘI - 2005

NHÀ XUẤT BẢN HÀ NỘI
4 - TỐNG DUY TÂN, QUẬN HOÀN KIẾM, HÀ NỘI
ĐIỆN THOẠI: (04)8.257063; 8.252916. FAX: (04)8.257063

GIÁO TRÌNH

CẤU TRÚC MÁY TÍNH

NHÀ XUẤT BẢN HÀ NỘI - 2005

Chịu trách nhiệm xuất bản:
NGUYỄN KHẮC OÁNH
Biên tập:
PHẠM QUỐC TUẤN
Bìa:
TRẦN QUANG
Trình bày - Kỹ thuật vi tính:
HOÀNG LAN HƯƠNG
Sửa bản in:
PHẠM QUỐC TUẤN

Mã số: 373 - 7.373 / HN - 05 107/407/05

In 1550 cuốn, khổ 17 x 24cm, tại Nhà in Hà Nội.
Giấy phép xuất bản số: 107GT/407 CXB ngày 29/3/2005
In xong và nộp lưu chiểu tháng 7 năm 2005.

Lời giới thiệu

Nước ta đang bước vào thời kỳ công nghiệp hóa, hiện đại hóa nhằm đưa Việt Nam trở thành nước công nghiệp văn minh, hiện đại.

Trong sự nghiệp cách mạng to lớn đó, công tác đào tạo nhân lực luôn giữ vai trò quan trọng. Báo cáo Chính trị của Ban Chấp hành Trung ương Đảng Cộng sản Việt Nam tại Đại hội Đảng toàn quốc lần thứ IX đã chỉ rõ: "Phát triển giáo dục và đào tạo là một trong những động lực quan trọng thúc đẩy sự nghiệp *công nghiệp hóa, hiện đại hóa*, là điều kiện để phát triển nguồn lực con người - yếu tố cơ bản để phát triển xã hội, tăng trưởng kinh tế nhanh và bền vững".

Quán triệt chủ trương, Nghị quyết của Đảng và Nhà nước và nhận thức đúng đắn về tầm quan trọng của chương trình, giáo trình đối với việc nâng cao chất lượng đào tạo, theo đề nghị của Sở Giáo dục và Đào tạo Hà Nội, ngày 23/9/2003, Ủyban nhân dân thành phố Hà Nội đã ra Quyết định số 5620/QĐ-UB cho phép Sở Giáo dục và Đào tạo thực hiện đề án biên soạn chương trình, giáo trình trong các trường Trung học chuyên nghiệp (THCN) Hà Nội. Quyết định này thể hiện sự quan tâm sâu sắc của Thành ủy, UBND thành phố trong việc nâng cao chất lượng đào tạo và phát triển nguồn nhân lực Thủ đô.

Trên cơ sở chương trình khung của Bộ Giáo dục và Đào tạo ban hành và những kinh nghiệm rút ra từ thực tế đào tạo, Sở Giáo dục và Đào tạo đã chỉ đạo các trường THCN tổ chức biên soạn chương trình, giáo trình một cách khoa học, hệ

thống và cập nhật những kiến thức thực tiễn phù hợp với đối tượng học sinh THCN Hà Nội.

Bộ giáo trình này là tài liệu giảng dạy và học tập trong các trường THCN ở Hà Nội, đồng thời là tài liệu tham khảo hữu ích cho các trường có đào tạo các ngành kỹ thuật - nghiệp vụ và đông đảo bạn đọc quan tâm đến vấn đề hướng nghiệp, dạy nghề.

Việc tổ chức biên soạn bộ chương trình, giáo trình này là một trong nhiều hoạt động thiết thực của ngành giáo dục và đào tạo Thủ đô để kỷ niệm "50 năm giải phóng Thủ đô", "50 năm thành lập ngành" và hướng tới kỷ niệm "1000 năm Thăng Long - Hà Nội".

Sở Giáo dục và Đào tạo Hà Nội chân thành cảm ơn Thành ủy, UBND, các sở, ban, ngành của Thành phố, Vụ Giáo dục chuyên nghiệp Bộ Giáo dục và Đào tạo, các nhà khoa học, các chuyên gia đầu ngành, các giảng viên, các nhà quản lý, các nhà doanh nghiệp đã tạo điều kiện giúp đỡ, đóng góp ý kiến, tham gia Hội đồng phản biện, Hội đồng thẩm định và Hội đồng nghiệm thu các chương trình, giáo trình.

Đây là lần đầu tiên Sở Giáo dục và Đào tạo Hà Nội tổ chức biên soạn chương trình, giáo trình. Dù đã hết sức cố gắng nhưng chắc chắn không tránh khỏi thiếu sót, bất cập. Chúng tôi mong nhận được những ý kiến đóng góp của bạn đọc để từng bước hoàn thiện bộ giáo trình trong các lần tái bản sau.

GIÁM ĐỐC SỞ GIÁO DỤC VÀ ĐÀO TẠO

Lời nói đầu

Trong những năm gần đây, sự phát triển mạnh mẽ của công nghệ thông tin làm gia tăng các nhu cầu học tập, nghiên cứu hệ thống máy tính. Sách mô tả hệ thống máy tính rất phong phú, tuy nhiên giáo trình dùng để giảng dạy hệ thống máy tính nói chung và cấu trúc máy tính nói riêng cho từng đối tượng có những đặc điểm khác nhau.

Cuốn giáo trình Cấu trúc máy tính *này biên soạn cho trình độ trung học chuyên nghiệp, trong đó tích hợp các kiến thức cơ bản nhất với một số phần tham khảo mở rộng.*

Mặc dù tác giả đã có nhiều cố gắng biên soạn với kinh nghiệm thực tế giảng dạy nhiều năm, cuốn giáo trình không tránh khỏi những thiếu sót. Tác giả mong nhận được các ý kiến đóng góp của độc giả.

Xin chân thành cảm ơn PGS.TS. Thái Quang Vinh - Viện Công nghệ thông tin, TS. Nguyễn Văn Điệp - Cao đẳng Điện lực đã đọc và cho những nhận xét về giáo trình. Xin trân trọng cảm ơn TS. Lương Cao Đông - Phó chủ nhiệm khoa CNTT Viện Đại học mở, TS. Lê Bá Dũng, TS. Lê Xuân Quảng - Viện Công nghệ thông tin, đã cho những ý kiến đóng góp quý báu để giáo trình hoàn thiện hơn.

Cuối cùng, xin chân thành cảm ơn Sở giáo dục và Đào tạo Hà Nội, Ban giám hiệu trường Trung học bán công kỹ thuật tin học Hà Nội (ESTIH) đã tạo điều kiện hoàn thành cuốn giáo trình này.

TÁC GIẢ

Bài mở đầu

Máy tính cá nhân ra đời đầu những năm 80 của thế kỷ 20 và ngày càng giữ một vị trí quan trọng trong các lĩnh vực khác nhau của xã hội và tạo ra bước tiến mới trong lịch sử nhân loại. Sự phát triển nhanh chóng của công nghệ máy tính, bao gồm cả máy tịnh cá nhân, làm cho việc mô tả nó gặp nhiều khó khăn, đòi hỏi phải luôn cập nhật. Giáo trình này được biên soạn cho học sinh các ngành Công nghệ thông tin và Điện tử viễn thông trong các trường trung học chuyên nghiệp. Nội dung chủ yếu của giáo trình nhằm trình bày những kiến thức về kiến trúc, cấu trúc máy tính cá nhân thông dụng và một số thiết bị ngoại vi thường gặp trong hệ thống máy tính cá nhân.

Trước đây quan điểm dạy học về máy tính ở nước ta có khác nhau: hoặc là dạy kiến trúc máy tính hoặc là dạy cấu trúc máy tính, dẫn đến việc xuất hiện hai loại giáo trình với tên gọi như trên. Giáo trình Kiến trúc máy tính chủ yếu cung cấp cho sinh viên các kiến thức về kiến trúc phân mức, trong khi hầu như các giáo trình Cấu trúc máy tính hoàn toàn không đề cập đến vấn đề này. Thiết nghĩ sinh viên cần phải hiểu kiến trúc phân lớp quan trọng của máy tính. Đồng thời phải nắm được những vấn đề căn bản, chi tiết của cấu trúc máy tính như biết nhận dạng, hiểu chức năng các chip có trong bản mạch chính, biết lập trình để vận hành các chip. Giáo trình này lấy kiến trúc phân mức của máy tính làm nền tảng và tổng hợp các cấu trúc chi tiết vào các mức tương ứng. Giáo trình gồm 6 chương.

Chương 1 - Nhập môn: Hệ thống lại các kiến thức về các hệ đếm khác nhau, đồng thời trình bày một cách ngắn gọn lịch sử phát triển máy tính và phân loại máy tính.

Chương 2 - Cấu trúc chung của máy tính: Trình bày cấu trúc chung của máy tính, nhấn mạnh cấu trúc mô phỏng con người, mô tả khái quát tất cả các

thành phần chủ yếu có mặt trong bản mạch chính cũng như tên gọi và chức năng của chúng trong một sơ đồ phối hợp tổng thể sát với thực tế.

Chương 3 - Các mức chương trình: Trình bày các mức máy tính. Mô tả ngắn gọn sự liên hệ giữa các mức, coi sản phẩm của mức này là nguyên liệu của mức trên, cũng như sự có mặt của mức trên là để phát triển, để khắc phục nhược điểm của mức dưới. Theo quan điểm đó, tại mức thiết bị sẽ trình bày việc thiết kế các sơ đồ logic cơ bản, các loại bộ nhớ. Tại mức số, trình bày việc thiết kế các sơ đồ logic tổng hợp từ các sản phẩm của mức thiết bị. Các sơ đồ này được điều khiển hoạt động bởi các từ điều khiển (các vi lệnh) và do đó có thể trình bày cách thức viết một đoạn vi chương trình tại mức vi chương trình. Mức hệ điều hành mô tả tiến trình khởi động của hệ điều hành thông dụng DOS như là một ví dụ. Các vấn đề quản lý bộ nhớ, quản lý tập tin... của hệ điều hành được phân tích trong các chương sau với một vi xử lý, một hệ điều hành cụ thể. Mức hợp ngữ thường là đối tượng của giáo trình Vi xử lý và vi điều khiển, vì vậy chỉ trình bày ngắn gọn một số lệnh hợp ngữ, cấu trúc chương trình và cách dịch một chương trình nguồn, nhằm giúp cho sinh viên có thể hiểu được một số đoạn chương trình gắn với phần cứng ở các phần sau. Mức ngôn ngữ bậc cao, vì là đối tượng giảng dạy của nhiều giáo trình khác, nên được mô tả chung trong phần tổng quan.

Chương 4 - Vi xử lý 8086: Trình bày chủ yếu cách thức quản lý bộ nhớ của hệ điều hành DOS với vi xử lý 8086. Phần mô tả tổng quát các vi xử lý thế hệ sau đề cập trong chương 2.

Chương 5 - Bộ nhớ ngoài: Trình bày chủ yếu cách quản lý bộ nhớ ngoài của hệ điều hành DOS, cũng như cấu trúc và tổ chức vật lý của các bộ nhớ này.

Chương 6 - Thiết bị ngoại vi và ghép nối: Trình bày cấu trúc của một số thiết bị ngoại vi thông dụng và cách thức ghép nối chúng với máy tính.

Đối với các sinh viên đã được học các giáo trình Kỹ thuật Điện tử và Kỹ thuật số phần thiết kế các cổng logic cơ bản ở mức thiết bị và toàn bộ mức số (thuộc chương 3), học sinh tự đọc để ôn lại, hệ thống lại các kiến thức về Kỹ thuật số cần cho môn Cấu trúc máy tính. Phần các chip hỗ trợ (thuộc chương 4), học sinh học và hiểu được khái niệm ngắt, phân loại ngắt, thủ tục ngắt, khái niệm và thủ tục DMA; các phần còn lại dành cho học sinh tự tham khảo có

hướng dẫn. Phần mã hoá và định dạng (thuộc chương 5) chỉ học phương pháp mã hoá FM, các phương pháp mã hóa còn lại học sinh tự đọc. Tóm lại, giáo trình này hướng tới việc trình bày một cách hệ thống các kiến thức cần thiết về cấu trúc và kiến trúc máy tính cho sinh viên. Tuy nhiên cần phải nhận thức rằng, nhà trường trang bị cho sinh viên một khối lượng kiến thức lớn, liên hoàn bổ sung lẫn nhau từ nhiều giáo trình. Trong một số trường hợp, nội dung của các giáo trình (được viết bởi các tác giả khác nhau) trùng nhau ở một số phần nhằm hệ thống kiến thức cho sinh viên, nhưng nếu phần trùng này lớn thì việc điều chỉnh nội dung giảng dạy là cần thiết.

Chương 1

NHẬP MÔN

Mục tiêu: Một cách tự nhiên, con người sử dụng hệ đếm thập phân (hệ đếm 10), do có mười ngón tay. Còn máy tính, được cấu tạo từ các mạch điện tử có hai trạng thái đóng và mở, chỉ hiểu và sử dụng hệ đếm nhị phân. Do vậy, trước khi nghiên cứu cấu trúc của máy tính, sinh viên phải hiểu được hệ đếm nhị phân, hệ 16, thực hiện được các phép toán cơ bản trong hệ đếm này, chuyển đổi được kết quả ra hệ thập phân và ngược lại. Trong chương này, ngoài những vấn đề trên, sau khi học xong sinh viên phải biết phân loại máy tính và nắm được lịch sử phát triển của máy tính cá nhân.

I. HỆ ĐẾM NHỊ PHÂN VÀ HỆ ĐẾM 16

1. Hệ thập phân (decimal)

Hệ thập phân là hệ đếm được nhân loại quen dùng. Người ta thường cho rằng, hệ đếm này bắt nguồn từ việc con người có mười ngón tay và từ xa xưa đã được sử dụng để đếm, để tính toán. Toàn thế giới thống nhất sử dụng mười ký tự Ai Cập : 0,1,2,3,4,5,6,7,8,9 để biểu diễn các con số của hệ thập phân, vì ưu điểm lớn nhất của hệ ký tự này so với hệ ký tự La mã hay Trung hoa ở chỗ có ký tự 0 và nhờ vậy việc biểu diễn các số lớn đơn giản hơn nhiều. Mỗi số nguyên trong hệ thập phân được tính theo công thức sau, ví dụ số $1953 = 1.10^3+9.10^2 +5.10^1 +3.10^0$, người ta nói cơ số của hệ thập phân là 10.

2. Hệ nhị phân (binary)

Hệ nhị phân bắt nguồn từ việc các mạch điện tử lôgic cấu tạo nên máy tính có hai trạng thái đóng và mở (không dẫn và dẫn điện). Người ta thiết kế sao cho khi mạch đóng thì ở lối ra của mạch có điện áp 5V, còn khi mở có điện áp 0V. Nếu gán cho điện áp 5V là số 1 và điện áp 0V là số 0, giống như gán cho 10

ngón tay là số 10, 9 ngón tay là số 9...thì ta có thể đếm các đồ vật, các sự kiện quanh ta bằng bộ đếm nhị phân xây dựng trên cơ sở các công tắc điện tử này. Môn học điện tử số và cấu trúc máy tính nghiên cứu thiết kế các vi mạch tổ hợp sao cho các mạch này có thể đếm được và tính toán được trong hệ nhị phân, sau đó chuyển đổi kết quả sang hệ thập phân cho con người dễ hiểu. Hệ nhị phân chỉ dùng hai con số là 0 và 1. Giá trị thập phân của một số nhị phân được tính theo công thức sau, ví dụ số $11010110 = 1.2^7 + 1.2^6 + 0.2^5 + 1.2^4 + 0.2^3 + 1.2^2 + 1.2^1 + 0.2^0 = 214$, cơ số của hệ nhị phân là 2. Mỗi con số nhị phân trong số nhị phân gọi là một bit (binary digit), bit được coi là đơn vị thông tin nhỏ nhất có thể lấy giá trị 0 hoặc 1 và tuỳ theo vị trí trong số nhị phân, bit có trọng số khác nhau. Trong ví dụ số nhị phân 8 bit nói trên, bit ngoài cùng bên trái có trọng số lớn nhất (most significant bit -MSB) và bit ngoài cùng bên phải có trọng số nhỏ nhất (least significant bit -LSB). Mạch điện tử khi nhớ hoặc truyền bit bằng 1(hay bằng 0) nghĩa là nhớ hoặc truyền mức thế bằng +5V hoặc 0V. Các bội số của bit như sau :

- 4 bit là một nibble.
- 8 bit là một byte.
- 16 bit là một từ (word).
- 32 bit là một từ kép (double word).

Các bội số lớn hơn của bit (b) hay byte (B) được gọi giống như trong hệ thập phân

-1 Kilobit (Kb) bằng 1024 bit hay bằng 2^{10} bit.

-1 Megabit (Mb) bằng 1024 Kb hay bằng 2^{20} bit.

-1 Gigabit (Gb) bằng 1024 Mb hay bằng 2^{30} bit.

-1 Terabit (Tb) bằng 1024 Gb hay bằng 2^{40} bit.

3. Hệ 16 (hệ thập lục phân hay hexadecimal)

Hệ 16 ra đời do việc tính toán, nhớ và viết số nhị phân rất khó và quá dài. Máy tính chỉ hiểu số nhị phân, do vậy các lệnh, các dữ liệu phải được viết dưới dạng nhị phân. Nhưng nhớ lệnh và dữ liệu dưới dạng các số nhị phân không đơn giản chút nào. Một số nhị phân 4 bit, tuỳ theo mỗi bit có giá trị 0 hay 1 sẽ có 16 tổ hợp có giá trị thập phân từ 0 đến 15, có thể biểu diễn số 4 bit này bằng một số hệ 16 và ta cần 16 ký tự. Người ta chọn 16 ký tự đó là 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. Do đó, thay vì nhớ một số nhị phân 4 bit người ta chỉ cần nhớ một ký tự 16 tương ứng, nếu số nhị phân lớn hơn 4 bit người ta

sẽ nhóm từng 4 bit một và tương ứng mỗi nhóm với một số hệ 16. Ví dụ số nhị phân 1010 0101 bằng số hệ 16 là A5 và ngược lại. Để phân biệt số hệ 16 với số thập phân, đặc biệt dễ nhầm khi số hệ 16 có chỉ các ký tự Ai Cập, người ta thường viết thêm chữ "h "vào sau số hệ 16. Giá trị thập phân của một số hệ 16 được tính theo công thức sau, ví dụ 2F1B h = $2.16^3 + 15.16^2 + 1.16^1 + 11.16^0$, cơ số của hệ thập lục phân là 16.

4. Cách chuyển đổi giữa các hệ đếm

Cách đổi số nhị phân và số hệ 16 ra số thập phân, số hệ nhị phân ra số hệ 16 và ngược lại đã trình bày ở phần trên. Trong phần này chỉ đề cập đến việc đổi số thập phân ra số nhị phân và số hệ 16. Để đổi ra hệ nhị phân hoặc hệ 16, người ta chỉ cần chia số thập phân cho cơ số của hệ 2 (hoặc 16). Số dư là trị số, thương số được chia tiếp để tính trị số tiếp theo, việc chia được tiếp tục cho đến khi kết quả bằng 0. Số dư đầu tiên là số LSB và số dư cuối cùng khi kết quả bằng 0 là số MSB. Ví dụ đổi số 113 ra hệ nhị phân và hệ 16 như sau :

Số nhị phân	Số hệ 16
113/2 = 56 dư 1 = LSB	113/16 = 7 dư 1
56/2 = 28 dư 0	7/16 = 0 dư 7
28/2 = 14 dư 0	
14/2 = 7 dư 0	
7/2 = 3 dư 1	
3/2 = 1 dư 1	
1/2 = 0 dư 1 = MSB	

Như vậy, số thập phân 113 = 1110001= 71h.

5. Biểu diễn thông tin theo mã nhị phân

Trong thế giới loài người, chúng ta không chỉ sử dụng các thông tin số, mà còn sử dụng các dạng thông tin khác như thông tin chữ (ký tự), thông tin hình ảnh, thông tin âm thanh. Trong khi đó máy tính chỉ dùng số nhị phân gồm hai ký tự 0 và 1. Để máy tính hiểu được các thông tin này, người ta phải mã hoá các thông tin phi số thành thông tin số. Ví dụ người Mỹ đã dùng bảng mã ASCII

(American standard code for information interchange) để mã hoá thông tin chữ và các ký tự điều khiển (có thể thấy trên bàn phím) bằng một số nhị phân 8 bit (xem bảng dưới).

II. PHÂN LOẠI MÁY TÍNH

Máy tính (computer) có nhiều loại, tuỳ theo khả năng, máy tính được chia thành các loại sau đây:

1. Máy tính lớn (mainfraime computer) có khả năng giải những bài toán lớn, tốc độ cao. Chúng có bộ nhớ rất lớn, bus dữ liệu rộng có thể hơn 64b, sử dụng nhiều vi xử lý, do vậy giá thành cao và được dùng cho những ứng dụng trong quân sự, hàng không, ngân hàng ...Ví dụ máy tính lớn là máy IBM4381.

Ký tự	Mã thập phân	Mã 16
NULL	0	00
31 ký tự điều khiển	1-31	01-1F
Các dấu	32-47	20-2F
Số 0-9	48-57	30-39
Các dấu khác	58-64	3A-40
A-Z	65-90	41-5A
Các dấu khác	91-96	5B-60
a-z	97-122	61-7A
Các dấu khác	123-127	7B-7F

2. Máy tính con (minicomputer) là dạng thu nhỏ của máy tính lớn cả về khả năng và kích thước, do vậy giá thành rẻ hơn và rất thích hợp trong nghiên cứu khoa học. Ví dụ về máy tính con là máy DEC,VAX 63600.

3. Trạm làm việc (workstation) là máy tính con hay máy vi tính, nhưng cho phép nhiều người có thể làm việc cùng một lúc thông qua mạng máy tính.

4. Máy tính cá nhân (personal computer-PC) chỉ cho phép một người sử dụng, cấu hình đơn giản và chuẩn hoá, được sản xuất hàng loạt, giá thành rẻ. Máy tính cá nhân có hai loại chính là để bàn (destop) và xách tay (notebook).

III. LỊCH SỬ MÁY TÍNH CÁ NHÂN

Lịch sử phát triển của máy tính cá nhân gắn liền với sự phát triển của máy tính IBM-PC.

Từ năm 1979 đến 1980: Hãng IBM hoàn thành thiết kế và cho ra đời máy Datamaster dùng vi xử lý 16 bit 8086 của hãng Intel. Máy này là cơ sở để phát triển máy tính cá nhân.

Năm 1980 máy tính cá nhân đầu tiên ra đời, đó là IBM-PC dùng vi xử lý 8 bit của Intel (8085).

Năm 1981-1982 máy IBM-PC/XT (extended technology) ra đời, sử dụng vi xử lý 8086 với 8 bit bus dữ liệu và 20 bit bus địa chỉ, tần số 4,77 MHz.

Năm 1984 máy PC/AT (advanced technology) ra đời, sử dụng vi xử lý 80286 của Intel có 16 bit bus dữ liệu và 24 đường địa chỉ, tần số đến 12 MHz.

Năm 1987 vi xử lý 32 bit 80386 ra đời, cùng với nó IBM cho ra đời máy tính 386 có 32 đường dữ liệu và 32 đường địa chỉ, tần số tới 40 MHz. Năm 1990 máy tính 486 ra đời có nhiều chức năng hơn 386, tần số tới 100 MHz, bus dữ liệu 64 bit.

Năm 1995 các máy tính đa phương tiện MMX, Pentium II với tần số 300 MHz xuất hiện và đến năm 1999 thì Pentium III ra đời. Đây là loại máy tính có khả năng biểu diễn không gian 3 chiều, nhận biết và tổng hợp được tiếng nói. Tần số làm việc điển hình của loại này lên tới 1,4 GHz.

Năm 2000 máy tính Pentium 4 với cấu trúc hoàn toàn mới ra đời, tần số làm việc lên tới hơn 2GHz. Năm 2002 dã xuất hiện vi xử lý siêu phân luồng làm việc làm tăng khả năng hoạt động đa nhiệm của máy tính, tần số làm việc trên 3GHz.

Bài tập chương 1

Bài 1: Đổi các số hệ 16 sau đây ra số thập phân và nhị phân: 7FE3h, A4CDh, EFD1h B15Ch.

Bài 2: Đổi các số nhị phân sau đây ra số hệ 16: 10110011, 11001010.

Bài 3: Đổi các số hệ thập phân sau đây ra số nhị phân và số hệ 16: 753, 198, 269, 875.

Bài 4: Phân loại các loại máy tính.

Bài 5: Mã ASCII của nhóm các chữ in và chữ thường từ A đến Z bằng bao nhiêu.

Chương 2

CẤU TRÚC CHUNG CỦA MÁY TÍNH

Mục tiêu : Máy tính nói chung và máy tính cá nhân nói riêng, được cấu trúc trên cơ sở mô phỏng con người. Đối tượng nghiên cứu của cuốn sách này là máy tính cá nhân PC/AT (personal computer / advanced technology). Các thành phần quan trọng nhất của PC/AT là bản mạch chính và vi xử lý - bộ não của máy tính. Học xong phần này sinh viên phải nắm được cấu trúc và nguyên tắc hoạt động của các bộ phận cơ bản của máy tính.

I. CẤU TRÚC MÔ PHỎNG CON NGƯỜI CỦA MÁY TÍNH

Để hiểu cấu trúc tổng quát và nguyên lý hoạt động chung của máy tính, ta xem xét cách xử lý tin của con người.

Con người dùng các cơ quan cảm thụ thông tin là các giác quan để nhận thông tin (ví dụ dùng mắt để nhận thông tin quang). Các thông tin này được đưa vào bộ nhớ ngắn hạn. Tại cơ quan tính toán quyết định, các thông tin vừa nhập được tính toán, cân nhắc so sánh với các thông tin cùng loại nhớ trong não, để đưa ra những quyết định xử lý thông tin đó. Do đó, con người có thể quyết định các hành động thích hợp và các quyết định hành động này được chuyển đến các cơ quan chấp hành để thực hiện hành động (ví dụ đến cơ quan phát âm để phát ra tiếng nói cần thiết). Cũng có thể các thông tin mới nhận sẽ được lưu vào bộ nhớ dài hạn, nếu chúng được coi là quan trọng. Để có thể điều phối hoạt động của các cơ quan trên, phải có một cơ quan điều khiển chung. Cơ quan này sẽ đưa ra các quyết định về thời điểm , loại thông tin cần thu nhập và chuyển tải đến cơ quan tính toán quyết định. Sau khi có quyết định, chính cơ quan này sẽ chuyển tải lệnh hành động cần thiết đến cơ quan chấp hành. Cơ quan điều khiển còn quyết định về các phép toán cần thực hiện ở cơ quan tính toán quyết định cũng như thời điểm đọc ghi thông tin với cơ quan nhớ thông tin (bộ nhớ). Đối

với con người, cơ quan nhớ, tính toán quyết định và điều khiển đều tập trung trong não người.

Mô phỏng con người, máy tính có bộ não là bộ vi xử lý, còn gọi là CPU (central proccessing unit). Trong bộ vi xử lý có bộ điều khiển (control unit) và bộ tính toán quyết định gọi là bộ số học logic ALU (arithmetic logic unit). ALU thực hiện các phép tính số học, các phép tính logic.

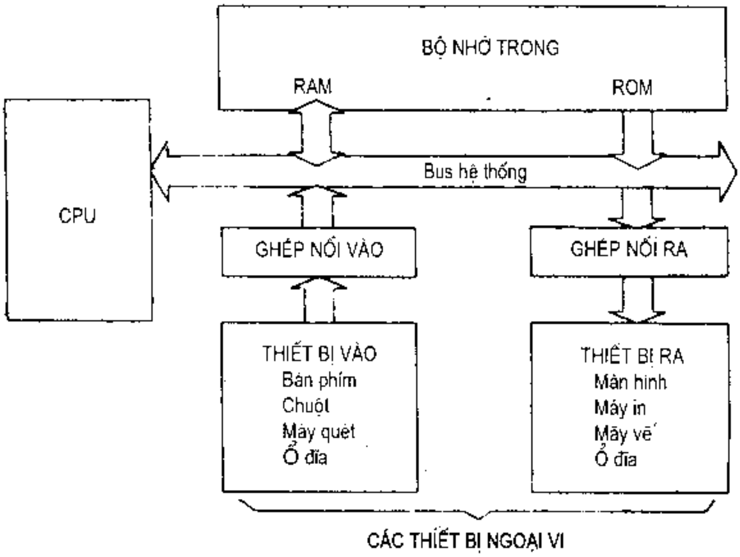
Khác với não người, bộ nhớ nằm ngoài vi xử lý, nhưng liên kết chặt chẽ với vi xử lý. Do với công nghệ hiện nay, bộ nhớ với dung lượng nhớ đủ lớn sẽ có kích thước lớn khó tích hợp vào bên trong vi xử lý. Bên cạnh đó, đặt bộ nhớ bên ngoài lại có ưu điểm cho phép người sử dụng tuỳ chọn kích thước bộ nhớ thích hợp với ứng dụng cụ thể, nhờ vậy mà tiết kiệm được kinh phí.

Tương đương với các cơ quan chấp hành của con người, máy tính có các thiết bị ngoại vi, còn gọi là các thiết bị vào ra I/O (input - output device). Các thiết bị này được nối với vi xử lý theo ba nhóm các dây song song dùng chung cho tất cả các thiết bị gọi là bus hệ thống. Như vậy bus hệ thống bao gồm ba nhóm và có tên lần lượt là : bus địa chỉ (bus A - address bus) dùng để truyền các thông tin địa chỉ, bus số liệu (bus D - data bus) dùng để trao đổi dữ liệu và bus điều khiển (control bus - bus C) để truyền các tín hiệu điều khiển và các thông tin về trạng thái thiết bị . Mỗi dây của bus truyền thông tin một bit, thông tin nhiều bit trao đổi giữa vi xử lý và các thiết bị ngoại vi được truyền song song trên các bus này. Danh sách các thiết bị vào ra thông thường đưa ra trong bảng sau:

Tên thiết bị ngoại vi	**Hướng trao đổi tin với vi xử lý**
Chuột (mouse)	Chỉ truyền tin vào (input device- I)
Card màn hình (monitor card)	Chỉ nhận tin ra (output device – O)
Bàn phím (keyboard)	I
Ổ đĩa cứng, đĩa mềm (hard disk, floppy disk)	I/O
Card âm thanh	O
Card mạng	I/O

Hình 2.1. Mô tả cấu trúc chung của máy tính

Do vi xử lý có kích thước nhỏ, nên không thể dành riêng cho mỗi thiết bị ngoại vi một bus riêng. Hơn thế nữa, nếu nối dây như vậy sẽ có rất nhiều tuyến dây đi từ vi xử lý và làm cho máy tính cồng kềnh, phức tạp, hoạt động không tin cậy.



Hình 2.1

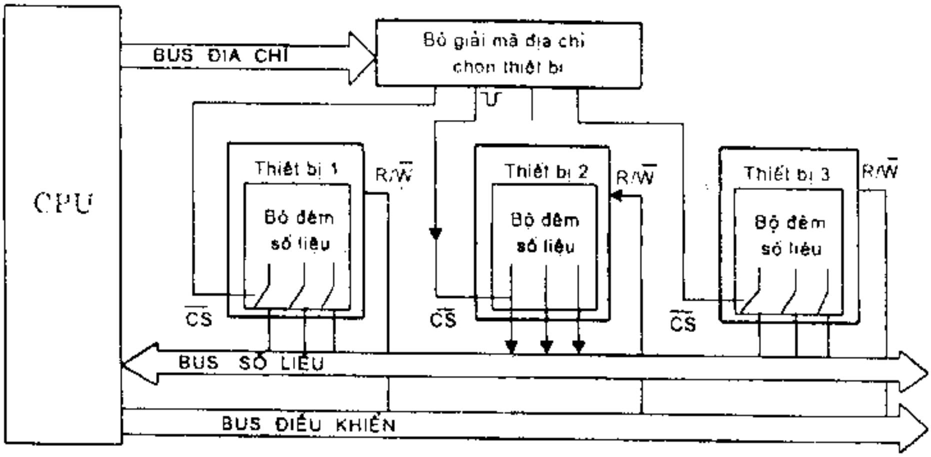
Vì thế, tất cả các thiết bị ngoại vi đều được nối về vi xử lý theo các bus chung. Tuy nhiên, khi đó để tránh trao đổi thông tin sai địa chỉ, vi xử lý trong một thời điểm chỉ làm việc với một thiết bị ngoại vi hay một ô nhớ. Để thực hiện điều đó, quá trình làm việc của vi xử lý với các thiết bị ngoại vi diễn ra theo các bước sau:

- *Bước 1:* Vi xử lý khi cần trao đổi thông tin với thiết bị ngoại vi nào sẽ phát địa chỉ của thiết bị đó theo mã nhị phân trên bus địa chỉ. Giả sử bus địa chỉ gồm 8 dây, như vậy có một số nhị phân 8 bit (8 con số nhị phân) được truyền trên bus này. Tám dây này (từ dây số 0 đến dây số 7) lần lượt có các trọng số là 2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7. Nếu trên dây nào đó đang truyền mức thế 5 vôn nghĩa là dây này đang truyền số 1, còn nếu đang truyền mức thế 0 vôn nghĩa là đang truyền số 0. Giả sử trên 8 dây đang truyền một số nhị phân 8 bit như sau: 00000010 , số nhị phân này có giá trị thập phân là 2 và giá trị hệ 16 là 2h. Điều đó có nghĩa là vi xử lý cần làm việc với thiết bị ngoại vi có địa chỉ thập phân là

2 (địa chỉ tính theo hệ 16 là 2h). Khi đó một bộ giải mã địa chỉ, có ở mỗi thiết bị ngoại vi, sẽ nhận được số nhị phân nói trên từ bus địa chỉ và so sánh với địa chỉ của thiết bị ngoại vi của mình. Nếu kết quả so sánh là đúng (giải mã so sánh) thì bộ giải mã địa chỉ sẽ phát tín hiệu để mở bộ đệm dữ liệu - phần nối giữa thiết bị ngoại vi số 2 và bus dữ liệu, còn bộ đệm dữ liệu của các thiết bị khác sẽ bị đóng lại, tức là các thiết bị khác bị ngắt khỏi bus dữ liệu chung và trên thực tế, tại thời điểm này chỉ có thiết bị ngoại vi số 2 được nối với bus D và vi xử lý.

- *Bước 2:* Vi xử lý trao đổi dữ liệu với thiết bị ngoại vi số 2 qua bus D.

Hình 2.2 trình bày sơ đồ nối dây giữa CPU và các thiết bị ngoại vi thông qua các bus.



Hình 2.2

Trong sơ đồ này, chỉ vẽ đại diện một bộ giải mã chọn địa chỉ. CPU phát tín hiệu địa chỉ của thiết bị cần làm việc số 2 lên bus địa chỉ. Tín hiệu này được đặt vào bộ giải mã địa chỉ và tạo ra tín hiệu cho phép mở chỉ bộ đệm dữ liệu của thiết bị 2 và chỉ có thiết bị này được trao đổi dữ liệu với vi xử lý qua bus dữ liệu. Còn trên bus điều khiển, trong ví dụ minh họa đơn giản này, chỉ truyền tín hiệu đọc/viết (R/$\overline{W}$) để chỉ chiều trao đổi dữ liệu:

- R/$\overline{W}$ = 1: vi xử lý đọc dữ liệu từ thiết bị ngoại vi hay dữ liệu được truyền từ thiết bị ngoại vi về vi xử lý.

- R/$\overline{W}$ = 0: vi xử lý viết dữ liệu vào thiết bị ngoại vi hay dữ liệu được truyền từ vi xử lý đến thiết bị ngoại vi.

Tương tự như với thiết bị ngoại vi, tại một thời điểm chỉ có một ô nhớ có địa chỉ xác định trên bus A được trao đổi dữ liệu với vi xử lý. Bộ nhớ gồm nhiều các ô nhớ, mỗi ô nhớ trong bộ nhớ được định địa chỉ tăng dần như số nhà. Để xác định vi xử lý tại thời điểm cho trước cần làm việc với thiết bị ngoại vi hay ô nhớ có cùng địa chỉ, người ta sử dụng thêm tín hiệu $IO/\overline{M}$ thuộc bus C. Khi $IO/\overline{M} = 1$ vi xử lý trao đổi dữ liệu với thiết bị ngoại vi, còn khi $IO/\overline{M} = 0$ vi xử lý chỉ trao đổi dữ liệu với bộ nhớ.

II. CẤU TRÚC CHUNG CỦA BẢN MẠCH CHÍNH

Vi xử lý, bộ nhớ chính và các chip, các khe cắm mở rộng, các mạch điện tử hỗ trợ cho hoạt động của vi xử lý đều được gắn trên bản mạch chính (mainboard) và liên kết với nhau thông qua các bus, các dây dẫn (mạch in) nằm trên chính bản mạch chính. Bản mạch chính được đặt trong một vỏ máy gọi là case (valy). Trên case còn lắp đặt các ổ đĩa cứng, ổ đĩa mềm và ổ đĩa CD (compact disk); các ổ này được nối về bản mạch chính bằng dây cáp tín hiệu và các dây cấp nguồn nuôi. Ngoài ra trên case còn có hộp cấp nguồn điện cho bản mạch chính và máy tính nói chung, gọi tắt là nguồn nuôi; nguồn nuôi nhận điện áp xoay chiều 220 vôn và biến đổi thành các điện áp một chiều cung cấp cho máy tính.

1. Vi xử lý

Phần trung tâm của bản mạch chính là vi xử lý. Vi xử lý thực hiện tất cả các hoạt động xử lý lôgic và số học. Nói chung, vi xử lý đọc số liệu từ bộ nhớ, xử lý các số liệu này theo hướng dẫn của lệnh (được viết bởi người lập trình) và sau đó viết kết quả vào bộ nhớ hoặc thiết bị ngoại vi. Cần phải nhấn mạnh ở đây một điểm khác biệt giữa con người và máy tính: con người có khả năng học và tự học nên có khả năng tự quyết định, tự ra lệnh để xử lý các dữ liệu nhận được. Còn máy tính mà bộ não của nó là vi xử lý không những cần dữ liệu, mà còn cần cả những lệnh để xử lý những dữ liệu đó; những lệnh này do người sử dụng cung cấp dưới dạng các chương trình.

Vi xử lý là một vi mạch tích hợp lớn, ví dụ vi xử lý Pentium 4 có khoảng bốn mươi triệu transistor. Có rất nhiều chân tín hiệu từ trong vi xử lý đưa ra ngoài để nối vi xử lý với bản mạch chính. Số lượng chân này thường là số hiệu của chân đế (socket) từ socket 370 trở đi. Tuỳ theo đời của vi xử lý mà có các loại chân đế khác nhau. Họ vi xử lý 8086 của hãng Intel có các loại chân đế sau:

- 80486 dùng chân đế loại socket 3 hoặc 8.
- 80586, MMX dùng chân đế loại socket 7.

- Pentium 2 hoặc 3 dùng socket 370 hoặc khe cắm mở rộng slot one.
- Pentium 4 dùng socket 423 hoặc 478.

Sở dĩ có các loại chân đế khác nhau vì việc nối dây, tạo chân cắm liên quan chặt chẽ đến độ tác động nhanh của vi xử lý. Nếu dây nối giữa các thành phần bên trong vi xử lý ngắn thì điện dung ghép nối nhỏ hơn, cho phép vi xử lý hoạt động ở tần số cao hơn.

2. Bộ nhớ chính

Trong khi tần số hoạt động của vi xử lý có thể đạt đến 3 GHz, thì người ta mới chế tạo được bộ nhớ chính hoạt động ở tần số thấp hơn nhiều. Bộ nhớ chính sử dụng loại chíp nhớ đọc/ghi được RAM (random access memory), được chia thành vài băng (bank). CPU dùng bộ nhớ để lưu trữ số liệu, các kết quả tính toán trung gian, các chương trình... Các dữ liệu được lưu trữ trong các đơn vị nhớ, gọi là ô nhớ (thường có kích thước tám bit gọi là một byte) và được sắp xếp trong bộ nhớ theo trật tự nhất định (ví dụ như được đánh số địa chỉ theo thứ tự tăng dần như đánh địa chỉ số nhà), nhờ vậy khi cần thiết có thể truy nhập được. Khi muốn đọc dữ liệu tại một ô nhớ có địa chỉ nào đấy, CPU phải truyền tín hiệu địa chỉ tới bộ nhớ theo bus địa chỉ, cho phép ô nhớ có địa chỉ đó được nối với bus D tương ứng từng bit một và dữ liệu trong ô nhớ được đọc ra và truyền vào vi xử lý. Thời gian kể từ khi CPU phát địa chỉ tới bộ nhớ cho đến khi dữ liệu có thể đọc được gọi là thời gian truy nhập bộ nhớ (access time). Tần số làm việc của bộ nhớ tỷ lệ ngược với thời gian này. Bộ nhớ chính thường sử dụng loại RAM động - DRAM (dynamic random access memory) mà yếu tố nhớ là tụ điện, do vậy thời gian đọc/ghi số liệu phụ thuộc vào thời gian phóng/nạp của tụ nên thường kéo dài, kết quả là tần số hoạt động chậm hơn các loại bộ nhớ khác. Có nhiều loại DRAM, bộ nhớ chính dùng SDRAM (synchronous DRAM) chỉ hoạt động được ở tần số 133 MHz, còn nếu dùng DDRAM (dual rate DRAM) thì tần số hoạt động được nhân đôi, tức là 266 MHz (sau này tần số có thể được nhân 3 hoặc hơn thế nữa, nhưng vẫn giữ nguyên tên gọi). Bộ nhớ có độ tác động nhanh cao hơn so với hai loại trên dùng RAMBUS hay RDRAM, có tần số hoạt động điển hình cỡ 533 MHz và 800 MHz tại thời điểm mà tần số hoạt động của vi xử lý đạt cỡ 3 GHz.

Bộ nhớ DRAM bao gồm từ một đến nhiều thanh RAM. Dung lượng của một thanh RAM có thể là 32, 64, 128, 256 Mbyte...Tuỳ theo thanh RAM được chế tạo theo chuẩn SIMM (single in-line memory module), DIMM (dual in-line

memory module) hay chuẩn cho DDRAM, RDRAM mà các thanh RAM được cắm vào khe cắm RAM chuẩn tương ứng trên bản mạch chính.

3. Các chip cầu nối hạ tốc độ

Các khối thiết bị ngoại vi, cùng với bản mạch chính để cấu thành PC/AT hoàn chỉnh, được nối về bản mạch chính thông qua các bản mạch ghép nối trung gian, còn gọi là card giao diện. Nhiều khi các bản mạch ghép nối trung gian này được tích hợp luôn vào bản mạch chính. Ví dụ như các ổ đĩa cứng và ổ đĩa mềm nối với bản mạch chính thông qua bản mạch IDE (card giao diện IDE, gọi tắt là card IDE), màn hình được ghép nối thông qua card màn hình, máy in được ghép nối thông qua mạch ghép nối song song LPT , chuột và modem được ghép nối thông qua mạch ghép nối nối tiếp COM...Vì thiết bị ngoại vi cũng có thể được ghép nối với bản mạch chính thông qua các card giao diện rời, không tích hợp trong bản mạch chính, nên trên bản mạch chính có lắp sẵn các khe cắm mở rộng chuẩn về cơ khí và điện. Bus hệ thống được nối ra các khe cắm mở rộng. Các card giao diện được cắm vào khe cắm mở rộng này và nhờ thế mà được nối với bản mạch chính và hệ thống. Tổ hợp thiết bị ngoại vi và card giao diện hay còn gọi là hệ thống thiết bị ngoại vi, tuỳ loại mà có thể làm việc ở các tần số khác nhau. Ví dụ như các thiết bị ngoại vi đặc biệt như là các thiết bị công nghiệp thường hoạt động ở tần số thấp, nên card giao diện của các thiết bị này thường dùng các khe cắm mở rộng chuẩn ISA (industrial standard architecture) với tần số hoạt động thấp 8,33 MHz. Các khe cắm mở rộng dành cho các thiết bị ngoại vi có tần số hoạt động cao hơn như là chuẩn PCI (peripheral component interconnect) với tần số hoạt động 33/66 MHz, chuẩn AGP (accelerated graphics port) với tần số hoạt động được tính theo đơn vị 4X (4x33MHz), 8X (8x33MHz), tuy nhiên vẫn thấp hơn nhiều so với vi xử lý

Như vậy, hiển nhiên cần phải hạ tốc độ của vi xử lý trước khi ghép nối với bộ nhớ hay hệ thống thiết bị ngoại vi. Do vậy, trên bản mạch chính, ngoài các linh kiện như vi xử lý, bộ nhớ chính, các chip hỗ trợ, các khe cắm mở rộng, còn có các chip cầu nối làm nhiệm vụ hạ tốc độ. Chip có nhiệm vụ hạ tốc độ của vi xử lý xuống tốc độ của DRAM và khe cắm mở rộng chuẩn AGP (có thể cả với khe cắm mở rộng PCI với các máy tính tốc độ thấp), thường được gọi là chip MCH (memory control hub) hoặc chip set, chip cầu bắc. Chip ICH (I/O control hub) hay chip cầu nam hạ tốc độ từ MCH đến tần số dùng cho chuẩn PCI, ISA và các chip hỗ trợ hoạt động ở tần số thấp.

4. Bộ nhớ đệm cache

Vi xử lý nối với bộ nhớ chính DRAM thông qua chip MCH theo tuyến dây được gọi là FSB (front side bus). Bus lối ra của chip này nối trực tiếp vào bộ nhớ chính DRAM. Do buộc phải dùng chip hạ tốc độ xuống tốc độ của bộ nhớ chính, nên tốc độ của hệ thống bị giảm nhiều so với tốc độ của CPU.

Để khắc phục điều đó, các máy vi tính còn được trang bị thêm các bộ nhớ đệm cache. Trên bản mạch chính, bộ nhớ cache thường nằm cạnh vi xử lý và là những vi mạch tích hợp nhỏ. Về cấu tạo, bộ nhớ cache là bộ nhớ dùng RAM tĩnh - SRAM (static RAM), không dùng yếu tố nhớ là tụ điện, mà dùng các mạch triger nên có thể làm việc ở tần số tương đương với tần số hoạt động của vi xử lý. Bộ nhớ cache nằm giữa và một đầu được nối trực tiếp vào vi xử lý theo một tuyến dây gọi là BSB (back side bus), còn đầu kia nối vào bộ nhớ chính. Vi xử lý trao đổi dữ liệu với bộ nhớ cache với tốc độ của vi xử lý và trong khi vi xử lý bận xử lý dữ liệu thì bộ nhớ cache trao đổi dữ liệu với bộ nhớ chính với tốc độ chậm của bộ nhớ chính. Giá thành của SDRAM cao hơn DRAM cho nên không được dùng để làm bộ nhớ chính. Do vậy, một cách hợp lý, kích thước bộ nhớ cache phải nhỏ. Điều này dẫn đến thông tin chứa trong bộ nhớ cache phải là thông tin chọn lọc và được dự đoán là vi xử lý sẽ cần đến trong lần trao đổi dữ liệu sau (ví dụ như lệnh tiếp theo hay dữ liệu vừa dùng đến). Kích thước của bộ nhớ cache nói chung cỡ một phần nghìn bộ nhớ chính (hiện nay đạt cỡ 512KB đến vài MB).

5. Bộ nhớ ROM-BIOS

Trên bản mạch chính còn có bộ nhớ cố định (hoặc bán cố định) ROM (read only memory), thường được phủ bằng giấy bạc ở mặt trên, nhằm phòng ngừa sai hỏng thông tin nhớ cố định ở bên trong vi mạch do tác động của sóng điện từ. Bộ nhớ ROM khác bộ nhớ RAM ở chỗ, thông tin trong bộ nhớ RAM sẽ mất đi khi mất nguồn cung cấp điện, còn với bộ nhớ ROM thì ngược lại. Máy tính chỉ hoạt động được khi bên cạnh phần cứng, máy còn được nạp các chương trình phần mềm điều khiển, trong đó, đầu tiên phải kể đến một tập hợp các chương trình của hệ điều hành (operating system). Hệ điều hành điều khiển, quản lý hoạt động của PC và tạo môi trường cho các chương trình phần mềm khác (ví dụ như các chương trình ứng dụng) chạy được. Hệ điều hành thường được làm hai phần và nạp sẵn vào hai bộ phận. Phần thứ nhất được nạp cố định vào ROM

và vi xử lý được thiết kế để chạy ngay phần này khi vừa bật máy. Phần thứ hai thường được nạp vào đĩa mềm hoặc đĩa cứng. Chương trình cuối cùng nằm trong ROM là chương trình tìm phần thứ hai của hệ điều hành, thường được qui ước nằm ở phần đầu tiên của đĩa mềm ổ A hoặc đĩa cứng khởi động. Vi xử lý sẽ nhận và chạy từng dòng lệnh của phần hệ điều hành nạp trong ROM và trên đĩa từ khi vừa bật máy, để khởi động máy tính. Quá trình khởi động máy bao gồm quá trình kiểm tra phần cứng của máy tính, nạp sẵn các chương trình quản lý máy tính vào bộ nhớ DRAM và đưa máy tính về trạng thái sẵn sàng chờ và thực hiện lệnh mới do người sử dụng đưa vào. Các chương trình quản lý máy tính chủ yếu sẽ kiểm tra, điều khiển hoạt động của các thiết bị ngoại vi, quản lý bộ nhớ, chính vì vậy người ta hay gọi ROM nói trên là ROM-BIOS (basic input- output system) và phần mềm trong ROM gọi tắt là BIOS. ROM được coi là một phần của bộ nhớ chính và được đánh địa chỉ cao nhất trong bộ nhớ qui ước (kích thước 1MB). Chương trình nạp cứng trong ROM có thể bị hỏng do nhiều nguyên nhân, khi đó có thể nạp lại các chương trình này nhờ truy cập vào trang Web của hãng sản xuất bản mạch chính.

6. RAM-CMOS

Một chip điện tử khác đáng chú ý trên bản mạch chính là RAM CMOS (RC). Khi lắp ráp máy tính, người ta có thể dùng các loại ổ cứng, ổ mềm khác nhau, có thể dùng cấu hình máy tính đầy đủ hoặc thiếu một bộ phận không thật quan trọng (ví dụ như ổ mềm). Đồng thời, người sử dụng có thể muốn cài đặt một số thông số (ví dụ như ngày, tháng, năm hoặc từ khoá bảo mật) hay quy định của riêng mình (như khởi động từ ổ đĩa cứng nào), để khi hoạt động máy tính phải tuân theo cấu hình đó. RC sẽ giúp người sử dụng làm được điều đó. RC là một bộ nhớ RAM làm từ linh kiện transistor loại CMOS và được cấp điện riêng biệt bằng một pin điện nạp được. Do vậy, khi tắt máy tính thông tin trong RC vẫn được giữ nguyên. Nhờ chương trình có sẵn trong BIOS (thường nhấn phím DEL khi khởi động để vào chương trình này), người sử dụng có thể đọc nội dung của RC, thay đổi cấu hình máy, hoặc nạp lại nội dung chuẩn cho RC từ BIOS. Như vậy có thể coi RC là một giao diện giữa người và máy (BIOS), cho phép người sử dụng thay đổi cấu hình máy trong phạm vi cho phép. Trong RC còn có một đồng hồ đo thời gian thực chạy bằng pin nói trên, cung cấp cho PC thời gian chính xác, gắn với các hoạt động của mình. Chip RC điển hình là chip MC 146818 có 64 B nhớ.

7. Chip I/O

Chip I/O nói chung là một chip lớn thứ ba trên bản mạch chính sau MCH và ICH. Giống như MCH, ICH ở cả bốn cạnh của vi mạch này đều có các chân cắm vào bản mạch chính. Nó hỗ trợ và là cầu nối thực hiện việc giao tiếp của vi xử lý (thông qua MCH và ICH) với một số thiết bị ngoại vi như bàn phím, giao tiếp nối tiếp, song song... Như vậy, chip I/O có bus lối vào nối với ICH và một số bus lối ra nối với một vài ngoại vi.

8. Các chip hỗ trợ

Trên đây vừa mô tả các thành phần, các chip điện tử chính có mặt trên bản mạch chính. Để hỗ trợ cho hoạt động của vi xử lý, trên bản mạch chính còn có các chip :

8.1. Chip hỗ trợ ngắt: Với máy tính PC đời đầu, một trong những vi mạch loại này có số hiệu 8259A, về sau này nó được tích hợp cùng với các vi mạch khác trong một chip lớn hơn (ví dụ như chip I/O). Ngắt (interrupt) là khả năng vi xử lý dừng chương trình chính đang chạy để chạy chương trình con phục vụ thiết bị ngoại vi phát tín hiệu đòi hỏi ngắt (đòi được phục vụ). Chip vi xử lý chỉ có một đầu vào (chân vi mạch) để nhận tín hiệu ngắt, nếu tín hiệu này bằng mức logic1 (tương đương mức điện thế 5 von) nghĩa là có yêu cầu ngắt gửi đến vi xử lý từ thiết bị ngoại vi, khi đó vi xử lý sẽ dừng (ngắt) chương trình đang chạy, để chạy chương trình con phục vụ thiết bị phát ra ngắt (ví dụ thiết bị đó là bàn phím) cho đến hết, rồi lại quay về thực hiện chương trình đang chạy dở. Cơ chế này mô phỏng khả năng có thể tạm thời ngắt việc này làm việc khác của con người. Chip 8259 có thể nhận tám tín hiệu ngắt từ tám thiết bị khác nhau, đầu ra của nó nối với đầu vào ngắt của vi xử lý và do đó giúp cho vi xử lý có khả năng phục vụ ngắt cho tám thiết bị ngoại vi một cách đồng thời.

8.2. Chip hỗ trợ thâm nhập bộ nhớ trực tiếp DMA (direct memory access): Một trong những vi mạch loại này có số hiệu 8237 và sau này cũng được tích hợp với các vi mạch khác trong một chip lớn hơn. Chế độ DMA và chip hỗ trợ tổ chức truyền trực tiếp một mảng dữ liệu giữa bộ nhớ và thiết bị ngoại vi nhờ phần cứng, không có sự điều khiển của vi xử lý (trung gian). Do vậy mà tốc độ truyền dữ liệu rất cao. Chip vi xử lý chỉ có một đầu vào để nhận yêu cầu DMA, nhờ chip 8237, vi xử lý có thể mở rộng khả năng nhận được bốn yêu cầu DMA cùng một lúc.

8.3. Chip định thời khả lập trình: Một trong vi mạch này có số hiệu 8253 hoặc 8254 và sau này cũng được tích hợp với các vi mạch khác trong một chip lớn hơn. Vi mạch này làm nhiệm vụ giống như đồng hồ số, nó nhận dao động điện từ bộ tạo dao động chuẩn thạch anh và chia tần để tạo ra các khoảng thời gian, các tần số khác nhau phục vụ cho hoạt động thường kỳ của máy tính. Ví dụ như các chu kỳ làm tươi bộ nhớ (hồi phục các tin lưu giữ trong bộ nhớ DRAM bị mất đi hay suy giảm do dò điện ở tụ nhớ), chu kỳ này cỡ chục mili-giây, tuỳ thuộc vào giá trị tụ làm bộ nhớ và dòng dò của tụ này. Nói chung, cứ sau 15 miligiây, xung vuông phát từ chip này sẽ tạo ra một chu kỳ làm tươi bộ nhớ (bằng cách đọc giả bộ nhớ). Ngoài ra chip này còn tạo khoảng thời gian (chu kỳ) để cập nhập đồng hồ hệ thống DOS, hoặc tạo ra các sóng âm với tần số khác nhau để phát ra loa...

9. Các card giao diện

Card giao diện có thể coi là một phần của thiết bị ngoại vi, dùng để cắm vào máy tính thông qua các khe cắm mở rộng chuẩn AGP, PCI hoặc ISA và từ đó được nối vào, trao đổi dữ liệu với vi xử lý. Card giao diện có thể được tích hợp ngay trên bản mạch chính, khi đó thiết bị ngoại vi được nối vào vào card thông qua các giắc cắm chuẩn nằm sau máy tính như LPT(line printer) cho các thiết bị trao đổi thông tin song song 8 bit, COM (communication) để trao đổi thông tin nối tiếp từng bit một và USB (universal serial bus) cho việc trao đổi thông tin cả gói, hoặc giắc cắm bàn phím... Trong mục này sẽ nghiên cứu khái quát một số card giao diện.

- Card giao diện ghép nối màn hình hay gọi tắt là card màn hình (CMH): CMH có nhiệm vụ nhận các thông tin hình ảnh và ký tự từ vi xử lý dưới dạng số, biến đổi chúng thành những thông tin thích hợp để điều khiển màn hình. Màn hình phổ biến nhất hiện nay là màn hình VGA (video graphic array). Màn hình VGA nhìn chung có chức năng và cấu tạo giống máy thu hình (tivi), do vậy cần những tín hiệu vào là những tín hiệu điều khiển màu dạng tương tự, tín hiệu đồng bộ... Do vậy, trên card màn hình có những bộ phận chủ yếu như: chip điều khiển, RAM video, bộ giải mã địa chỉ, bộ biến đổi tín hiệu số thành tín hiệu tương tự (DAC)...Đặc biệt, trên card còn có một ROM-BIOS riêng, chứa phần mềm hỗ trợ cho ROM-BIOS trên bản mạch chính, gồm các chương trình con dùng để tắt bật các chế độ hiển thị khác nhau hoặc dùng các trang trong bộ nhớ video... Ở phía sau bản mạch chính là một đầu nối gồm ba hàng chân

(connector) dùng để nối CMH với màn hình.

- Mạch ghép nối song song và nối tiếp: Sở dĩ gọi là mạch vì card giao diện loại này không tích hợp thành một card riêng biệt mà nằm ngay trên bản mạch chính. Mạch ghép nối có một đầu nối với bus hệ thống của vi xử lý, một đầu nối với các thiết bị nhận tin song song thường là 8 bit (máy in) hoặc thiết bị nhận tin nối tiếp 1 bit như là Modem... Bus hệ thống của vi xử lý là đường truyền có thể lên tới 64 bit. Như vậy, trong card giao diện này có những bộ nhận tin song song nhiều bit và biến đổi thành tin nối tiếp hoặc tin song song 8 bit ở đầu ra và ngược lại. Tin nối tiếp thường được đưa qua một đầu nối gọi là cổng COM, còn tin song song 8 bit được đưa qua cổng LPT nằm phía sau máy tính. Ngoài ra, giống như các card giao diện khác, mạch ghép nối phải có các bộ giải mã địa chỉ và các mạch điều khiển.

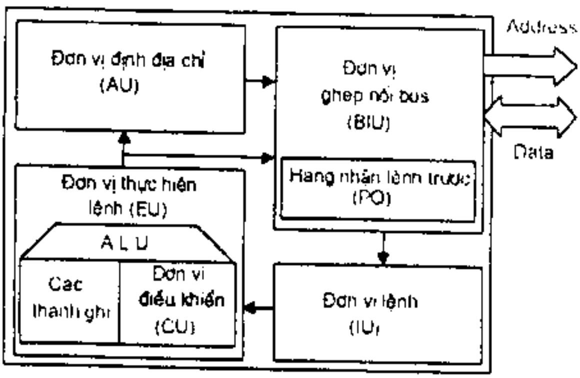
10. Nguồn nuôi cho CPU

Điện áp lưới xoay chiều 220V/110V được biến đổi thành các điện áp một chiều âm dương đối xứng 5V, 12V, 3,3V bởi một bộ biến đổi nguồn điện có ổn áp xung (switching) đựng trong một hộp nguồn gắn trên khung vỏ máy tính. Các điện áp một chiều này cấp cho bản mạch chính. Trên bản mạch chính có những đầu nối (giắc hay connector) để nhận các điện áp này. Thường thì có hai loại giắc AT hay ATX. Giắc AT là giắc12 chân, được nối với hai giắc thành phần từ hộp nguồn thường ký hiệu P8, P9, cho phép nhận điện áp nuôi thụ động từ hộp nguồn. Giắc ATX là giắc 20 chân còn cho phép truyền các tín hiệu điều khiển, tín hiệu trạng thái giữa bản mạch chính và hộp nguồn, phục vụ cho việc tắt bật nguồn nuôi hoặc đưa nguồn nuôi về trạng thái chờ công suất thấp (stand-by) bằng phần mềm hoặc thông qua bản mạch chính. Trên bản mạch chính, ngoài việc quan sát thấy các giắc này, còn có các mạch biến đổi điện áp một chiều lấy từ hộp nguồn thành các điện áp thấp hơn cấp cho vi xử lý (từ 1,6V đến 3,3V). Mạch này thường gồm các transistor công suất, tụ lọc, cuộn cảm... Sau này việc nguồn nuôi cho vi xử lý được cung cấp trực tiếp từ hộp nguồn. Nói chung vi xử lý càng hiện đại thì điện áp nguồn nuôi càng thấp để giảm công suất toả nhiệt khi dòng tiêu thụ tăng.

III. CẤU TRÚC CHUNG CỦA VI XỬ LÝ

Có nhiều hãng sản xuất vi xử lý như Intel, Cyrix, AMD, Philip ..., sản phẩm gồm hai dòng là vi xử lý và vi điều khiển (một chip tích hợp vi xử lý, bộ nhớ,

cổng I/O). Ba hãng đầu chủ yếu sản xuất vi xử lý. Cấu trúc chung của các vi xử lý hiện đại khá giống nhau. Để làm ví dụ, chúng ta nghiên cứu vi xử lý 8086 của Intel, đó là vi xử lý đầu tiên và có cấu trúc điển hình cho họ vi xử lý 8086 và Pentium. Các vi xử lý hiện đại sau này nhờ áp dụng các kỹ thuật màng mỏng để chế tạo transistor có kích thước ngày càng mỏng nên tần số hoạt động ngày càng cao; dung lượng các thanh ghi (bộ nhớ ngắn hạn) ngày càng lớn, cho phép xây dựng bus hệ thống có khả năng đồng thời truyền nhiều bit hơn. Các kỹ thuật đường ống (pipe line) được áp dụng rộng rãi cùng với việc phân luồng dữ liệu làm cho vi xử lý thực hiện được nhiều việc cùng lúc, nhờ vậy mà nâng cao tốc độ thực tế. Số lượng các đoạn vi chương trình nạp trong vi xử lý ngày càng lớn làm cho tập lệnh của vi xử lý ngày càng phong phú, nhờ thế vi xử lý trở lên mạnh hơn. Tuy nhiên, việc nghiên cứu vi xử lý thuộc thế hệ đầu tiên 8086, cung cấp các kiến thức căn bản cho phép việc nghiên cứu các vi xử lý sau này thuận lợi hơn.



Hình 2.3

Trước khi nghiên cứu sơ đồ khối (Hình 2.3), cần phải nhấn mạnh một lần nữa rằng: vi xử lý là bộ óc của máy tính, nó thực hiện mọi hoạt động xử lý lôgic và số học, nói chung bằng cách: đọc số liệu từ bộ nhớ, xử lý các số liệu này theo cách được xác định, được hướng dẫn bởi lệnh và sau đó viết kết quả vào bộ nhớ hoặc thiết bị ngoại vi.

Sơ đồ khối của vi xử lý 8086 gồm đơn vị ghép nối bus BIU (bus interfaced unit), đơn vị định địa chỉ AU (addressing unit), đơn vị lệnh IU (instruction unit) và đơn vị thực hiện lệnh EU (executive unit). EU gồm bộ tính toán số học-logic

ALU (arithmetical-logical unit), bộ điều khiển CU (control unit) và các thanh ghi. Quá trình làm việc diễn ra như sau :

AU dựa vào kết quả tính toán từ EU sẽ truyền địa chỉ thực chuẩn hoá (formated) cần truy nhập vào bus địa chỉ (bus A) thông qua BIU. Như vậy, địa chỉ của ô nhớ hay thiết bị ngoại vi được trao đổi dữ liệu với vi xử lý đã được xác định. Giả sử đó là quá trình đọc dữ liệu vào vi xử lý, theo bus dữ liệu (bus D), dữ liệu được nạp vào hàng nhận lệnh trước PQ (prefetch queue), mà thực chất là bộ nhớ nhỏ có dung lượng 6 byte (6B). Dữ liệu là lệnh, sẽ được giải mã lệnh bằng IU thành các từ điều khiển gồm các bit điều khiển có giá trị 0 (0V) hoặc 1 (5V). Các bit và từ điều khiển này sẽ điều khiển phần cứng nằm trong EU thực hiện lệnh. Ví dụ, nếu lệnh là lệnh cộng hai toán hạng có trong lệnh thì các bit điều khiển thông qua CU điều khiển bộ cộng trong ALU thực hiện lệnh đó và kết quả chứa vào thanh ghi nào đó (thường là AX) theo chỉ dẫn của lệnh. Như vậy thanh ghi là bộ nhớ tạm có dung lượng nhỏ (thường là 2B) nằm ngay trong vi xử lý để chứa các kết quả trung gian hay kết quả cuối cùng trong quá trình vi xử lý thực hiện chương trình.

Quá trình xử lý dữ liệu của vi xử lý rõ ràng có hai giai đoạn độc lập: lấy dữ liệu từ bộ nhớ và giải mã, thực hiện lệnh, nếu thực hiện hai công đoạn này tuần tự thì lãng phí thời gian. PQ cho phép lấy một vài lệnh vào vi xử lý, như vậy, khi lệnh thứ nhất được lấy vào PQ và đang được giải mã thực hiện, cùng lúc đó lệnh thứ hai sẽ được lấy sẵn vào, nhờ thế sau khi xử lý xong lệnh thứ nhất vi xử lý hầu như không phải mất thời gian lấy lệnh thứ hai. Hai đơn vị EU và BIU có thể làm việc độc lập với nhau. Đây là cấu trúc xử lý lệnh theo đường ống (pipeline) nhằm nâng cao độ tác động nhanh của vi xử lý. Tuy nhiên, điều cần lưu ý ở đây nếu lệnh thứ nhất là lệnh nhảy (JUMP=JMP) thì cần phải xử lý xong lệnh thứ nhất mới biết được địa chỉ của lệnh tiếp theo. Điều này được khắc phục bởi các vi xử lý đời sau.

Vi xử lý có một số thông số kỹ thuật đáng quan tâm:

- Đặc trưng tần số: Thường người sử dụng quan tâm đầu tiên đến đặc trưng này của vi xử lý. Một vi xử lý có khả năng hoạt động ở tần số càng cao thì thời gian xử lý lệnh hay thời gian để chạy một chương trình càng ngắn. Vi xử lý, nhìn một cách tổng quát, là một số lượng lớn các transistor hoạt động trong các sơ đồ tổ hợp khác nhau, nó nhận các thông tin "0" hoặc "1" ở lối vào và sau một thời gian trễ sẽ cho các thông tin kết quả ở lối ra. Bản chất của các thông tin "0"

và"1" là các mức điện áp 0V và 5V một cách tương ứng. Các mức điện áp này tác động vào transistor của một sơ đồ nào đó và cho ra mức điện áp 0V hoặc 5V tuỳ theo hàm mà sơ đồ đó thực hiện. Transistor có một quán tính nhất định vì trong cấu hình tương đương của nó có thành phần tụ điện nằm ở lối ra, thường thì để có mức 5V ở lối ra tụ này nạp điện, còn muốn về mức 0V tụ này phải phóng điện. Như vậy muốn có thông tin ở lối ra thì phải đợi tụ phóng và nạp đến giá trị điện áp cần thiết. Tụ có giá trị càng lớn, điện tích có trên tụ càng nhiều (Q=C.U) và do vậy phóng càng lâu và vì thế quán tính của transistor hay thời gian trễ của thông tin lối ra so với thời điểm tác động tin lối vào càng lớn. Khi đó ta nói sơ đồ hoạt động chậm hoặc tần số hoạt động của vi xử lý thấp. Để tăng độ tác động nhanh của vi xử lý cần phải chế tạo các transistor thành phần và các linh kiện mỏng, nhỏ. Khi đó các transistor và mạch điện có điện dung lối ra và điện dung lắp ráp nhỏ. Vi xử lý Pentium 4 được chế tạo nhờ công nghệ màng mỏng 0,13 μm, do vậy tần số làm việc đạt tới vài GHz.

- Đặc trưng công suất : Pentium 4 có khoảng 40 triệu transistor, nếu sử dụng transistor thông thường (lưỡng cực) tiêu thụ dòng cỡ 10^{-3} A thì tổng dòng tiêu thụ cỡ chục nghìn ampe, như vậy quá lớn. Người ta phải sử dụng các transistor MOS tiêu thụ dòng nhỏ. Nếu dòng này cỡ 10^{-6} A thì tổng dòng tiêu thụ là 40 A. Có thể ước lượng rằng, trong 40 triệu transistor ấy, một nửa đang có mức 1 ở lối ra, còn một nửa ở mức 0, hay nói một cách khác, có nửa số transistor đang đóng không tiêu thụ dòng, khi ấy tổng dòng tiêu thụ còn 20A. Điện áp nuôi cho vi xử lý cỡ 1,7 V thì công suất tiêu thụ là 34W có thể còn làm nóng, ảnh hưởng tới hoạt động vi xử lý. Biện pháp hữu hiệu để giải quyết vấn đề này là dùng quạt toả nhiệt.

Bài tập chương 2

Bài 1: Vẽ sơ đồ khối của bản mạch chính bao gồm các chip: vi xử lý, MCH...

Bài 2: Tốc độ hệ thống máy tính phụ thuộc vào những yếu tố, linh kiện nào.

Bài 3: Vẽ sơ đồ khối nối thiết bị ngoại vi với vi xử lý.

Chương 3

CÁC MỨC CHƯƠNG TRÌNH

Mục tiêu: Máy tính có thể hiểu bao gồm phần cứng, phần mềm và phần mềm nhúng trong phần cứng (phần sụn). Tuy nhiên chi tiết hơn, có thể coi máy tính gồm 7 mức. Người nghiên cứu có thể chọn lĩnh vực nghiên cứu của mình trong một hoặc vài mức, nhưng nhất thiết phải hiểu được sự liên hệ giữa các mức này. Các mức này tạo nên kiến trúc phân lớp của máy tính. Học xong chương này, sinh viên phải nắm được kiến trúc phân lớp nói trên.

I. TỔNG QUAN VỀ CÁC MỨC MÁY TÍNH

Máy tính thường được coi là có ba phần chính: Phần cứng (hardware), phần mềm (software) và phần sụn (firmware).

Phần cứng bao gồm các đối tượng hữu hình như các vi mạch (IC), các bảng mạch in, cáp nối, nguồn điện, bộ nhớ, máy đọc bìa, máy in dòng và terminal chứ không phải là các ý tưởng trừu tượng, các thuật toán hay là các chỉ thị.

Phần mềm (software) thì trái lại, nó bao gồm các thuật toán và các biểu diễn cho máy tính của chúng, đó chính là các chương trình. Chương trình có thể được lưu trữ trên bìa đục lỗ, trên băng từ, đĩa từ và các môi trường khác, tuy nhiên cái cơ bản nhất của phần mềm chính là tập các chỉ thị tạo nên chương trình chứ không phải là môi trường vật lý được sử dụng để ghi chương trình.

Phần sụn hay phần nhão (firmware) là một dạng trung gian giữa phần cứng và phần mềm, nó là phần mềm được nạp vào, nhúng vào các mạch điện tử trong quá trình chế tạo các mạch điện tử này. Firmware được sử dụng khi các chương trình hiếm khi hoặc không bao giờ cần phải thay đổi, thí dụ trong các đồ chơi hoặc các dụng cụ, máy móc. Firmware cũng được sử dụng khi các chương trình không được phép bị mất đi khi tắt điện. Trong nhiều máy tính, các vi chương trình thuộc firmware.

Nhưng chi tiết hơn, có thể coi máy tính gồm bảy mức:

1. Mức –1 (level –1 hay L-1) là mức thiết bị. Sở dĩ gọi là mức –1 mà không gọi là mức 0, chỉ số đầu tiên thường dùng, vì sau này ít đề cập đến mức này trong sơ đồ kiến trúc máy tính. Tại mức này người thiết kế nắm vững nguyên tắc hoạt động và nguyên tắc thiết kế các mạch điện từ các linh kiện cơ bản như transistor, điôt, điện trở, tụ điện, cuộn cảm... Đặc biệt là việc thiết kế các sơ đồ logic cơ bản như And, Or, Not (hệ hàm đủ) và bộ nhớ cho mức trên. Các sơ đồ And, Or, Not có một số lối vào, nếu đưa vào đầu vào các sơ đồ này các tín hiệu số (tín hiệu biểu diễn các số 0,1), nó sẽ cho một hàm logic cơ bản ở lối ra (tức là cho kết quả cũng là số 0 hoặc 1 ở lối ra tuỳ vào hàm và biến logic hay tín hiệu đầu vào). Bộ nhớ là sơ đồ điện tử có thể lưu giữ các số 0 hoặc 1 theo thời gian, cũng như có thể viết thông tin 0 hoặc 1 vào nó.

2. Mức 0 (level 0 hay L0) là mức logic số. Trên cơ sở các sơ đồ logic cơ bản And, Or, Not (còn gọi là hệ hàm đủ) hoặc các sơ đồ cơ bản tương đương (ví dụ như chỉ And và Not) tạo ở mức thiết bị, người thiết kế ở mức này sẽ có đủ "vật liệu" để thiết kế các sơ đồ logic tổ hợp và mạch dãy phức tạp hơn như là bộ cộng- trừ hai số nhị phân, bộ giải mã, bộ so sánh, bộ số học-logic ALU, các bộ chuyển mạch điện tử, bộ đếm, bộ ghi dịch... Một điều cần nhấn mạnh ở đây là khái niệm hệ hàm đủ. Các sơ đồ And, Or, Not là hệ hàm đủ vì chỉ bằng các sơ đồ này có thể thiết kế tất cả các sơ đồ logic phức tạp hơn, tích hợp lớn hơn. Người ta bắt đầu điều khiển được hoạt động của các sơ đồ tích hợp lớn nói trên bằng việc truyền các bit điều khiển, (còn gọi là từ điều khiển hay vi lệnh) đến lối vào điều khiển của sơ đồ.

Mức –1 và mức 0 là phần cứng của máy tính.

3. Mức 1 (level 1 hay L1) là mức vi chương trình (microprogramming level). Vi xử lý bao gồm một số vi mạch tích hợp lớn như bộ số học-logic ALU, giải mã, điều khiển, các bộ chuyển mạch điện tử, các thanh ghi, thanh ghi 3 trạng thái... Chúng ta cần phải điều khiển cho các sơ đồ này hoạt động nhịp nhàng nhằm thực hiện một chỉ thị hay một lệnh máy (lệnh máy thuộc mức sau là mức 2). Vi chương trình hay cụ thể một đoạn vi chương trình bao gồm một số các từ điều khiển như đề cập ở phần trên, được dùng để điều khiển phần cứng của vi xử lý thực hiện một nhiệm vụ, một hàm chỉ ra bởi lệnh máy. Ví dụ như lệnh cộng hay trừ hai số nhị phân với nhau, hai số nhị phân này đã được lấy từ bộ nhớ vào vi xử lý nhờ các lệnh chuyển số liệu trước đó. Như vậy mức 1, khác với mức 0, là ngôn ngữ máy với khái niệm chương trình là một dãy các vi lệnh (các từ điều khiển). Có thể nói qui trình làm việc của máy tính diễn ra như sau: Người lập trình viết chương trình bằng lệnh máy nhị phân. Lệnh máy này được

định nghĩa trong một bộ lệnh của mỗi loại vi xử lý, giả sử cần có một tập lệnh nhị phân khoảng 256 lệnh thì rõ ràng phải dùng số 8 bit để mã hoá các lệnh này và người ta có thể qui định, ví dụ lệnh máy 0000001 là lệnh cộng hai số nhị phân hay 00000010 là lệnh nhân hai số nhị phân... Chương trình viết bằng lệnh máy nhị phân được nạp vào bộ nhớ của máy tính. Vi xử lý sẽ lấy vào từng lệnh để từng bước thực hiện chương trình nói trên. Mỗi một lệnh máy sẽ được đổi (còn gọi là được giải mã hay thông dịch) thành một đoạn vi chương trình và chính đoạn vi chương trình này mới thực sự chạy để thực hiện chức năng (hàm) qui định bởi lệnh máy. Vi chương trình được nạp trong bộ nhớ cố định (control store) nằm bên trong IU của vi xử lý. Nói chung, mỗi lệnh máy ứng với một đoạn vi chương trình, có bao nhiêu đoạn vi chương trình thì có bấy nhiêu lệnh máy. Ví dụ vi xử lý chỉ có 16 đoạn vi chương trình, 16 đoạn này sẽ điều khiển phần cứng thực hiện 16 nhiệm vụ, hàm khác nhau, do đó vi xử lý này sẽ chỉ có bộ lệnh gồm 16 lệnh máy và chỉ dùng một số 4 bit với 16 tổ hợp các số nhị phân 0 và 1 để ký hiệu (mã hoá) là đủ. Vi chương trình được gọi là phần sụn (firm ware) của máy tính hay còn gọi là phần mềm nhúng trong phần cứng, vì vi chương trình thực sự là phần mềm được nạp trong một chip vi mạch cứng (control store).

4. Mức 2 (level 2 hay L2) gọi là mức lệnh máy hay mức lệnh máy nhị phân. Như đã phân tích ở trên, lệnh máy do người lập trình viết không thực sự điều khiển phần cứng thực hiện nhiệm vụ, hàm qui định bởi lệnh. Lệnh máy phải được đổi, được thông dịch thành đoạn vi chương trình tương ứng, gồm các vi lệnh và chính đoạn vi chương trình này mới thực sự điều khiển phần cứng thực hiện lệnh. Như vậy, lệnh máy là ảo và nó chỉ có giá trị thực sự khi gắn với vi chương trình. Tuy nhiên, chúng ta không thể lập trình bằng cách viết đoạn vi chương trình này nối tiếp đoạn khác. Lệnh máy cho mỗi đoạn vi chương trình một cái tên, một ký hiệu lệnh và làm cho việc lập trình trở nên thú vị, dễ dàng hơn và thực sự là một ngôn ngữ (L2). Nhưng chính lệnh máy nhị phân gồm một dãy các số 0 và 1 rất khó nhớ, người ta thường dùng số hệ 16 để ký hiệu lệnh ngắn gọn và dễ nhớ hơn. Ví dụ như một lệnh 8 bit viết dưới dạng nhị phân là 0110 1110 thì dạng 16 tương đương sẽ là 6Eh, dễ nhớ hơn nhiều. Sau này, người ta thấy ngay cả dùng hệ 16 cũng khó nhớ công dụng của một mã lệnh và đã sử dụng các ký tự dễ nhớ và gợi nhớ để ký hiệu lệnh, ví dụ nếu chuỗi số nhị phân trên là lệnh cộng thì lệnh này được ký hiệu là Add (từ tiếng Anh nghĩa là cộng).

Cả lệnh và dữ liệu đều là số nhị phân mà PC có thể lưu trong bộ nhớ và nhập vào CPU qua bus số liệu để xử lý. Lệnh được nhập vào bộ giải mã lệnh, còn dữ

liệu được đưa đến thanh ghi số liệu. Một lệnh máy gồm các phần như sau:

Tiền tố	Mã toán	Các toán hạng	Địa chỉ lệnh kế tiếp

Phần trung tâm là mã thao tác hay mã toán (operation code - opcode) bao gồm một hoặc hai byte. Hai bit bậc thấp của mã toán thường xác định hướng truyền số liệu (từ thanh ghi tới bộ nhớ hoặc từ bộ nhớ tới thanh ghi...) cũng như sử dụng các toán hạng 8 hay 16 bit (ví dụ AL hay AX của vi xử lý 8086). Toán hạng (operand) dùng để xác định những đối tượng ở đó phép toán được thực hiện, chúng có thể là nội dung các thanh ghi, các giá trị tức thời hay các địa chỉ của các giá trị hay địa chỉ của các địa chỉ các giá trị. Toán hạng có thể có, có thể không, thí dụ lệnh cộng hay nhân hai số hạng với nhau yêu cầu 2 toán hạng; lệnh dịch (shift) chỉ cần một, trong khi đó lệnh xoá cờ mang lại không cần toán hạng nào cả. CPU sẽ tìm lệnh kế tiếp tại địa chỉ lệnh kế tiếp. Phần này cũng có thể có, có thể không. Ví dụ khi dùng phương pháp địa chỉ hiểu ngầm qua thanh ghi con trỏ lệnh (ví dụ IP của 8086) thì không cần địa chỉ lệnh kế tiếp, nhưng trong các lệnh rẽ nhánh (JMP) sẽ phải chỉ ra rõ ràng địa chỉ lệnh kế tiếp... Trong một số trường hợp đặc biệt có thể có thêm phần tiền tố (prefix) nữa, ví dụ với 8086, tiền tố 3Eh báo vô hiệu đoạn DS chẳng hạn.

5. Mức 3 (level 3 hay L3) là mức máy hệ điều hành (operating system machine). Hệ điều hành dùng để quản lý máy tính như quản lý bộ nhớ, quản lý các thiết bị I/O, quản lý tập tin, kiểm tra phần cứng máy tính, đưa máy tính về trạng thái sẵn sàng thực hiện lệnh mức 3 (các file chạy được cũng được coi là lệnh mức 3), cũng như tạo môi trường cho các lệnh này chạy được. Hệ điều hành luôn gồm hai phần. Phần chương trình chạy ngay khi bật máy, do vậy phải viết bằng lệnh máy nhị phân và được chứa trong chip ROM - BIOS. Phần thứ hai thường được chứa trên đĩa từ hay đĩa quang bao gồm một số file hệ thống và tập lệnh ký tự. Do vậy mức 3 là một mức lai, hầu hết các chỉ thị (lệnh) có trong ngôn ngữ của nó (L3) cũng có trong ngôn ngữ mức 2 (L2). Các lệnh ký tự (ví dụ như tập lệnh ký tự của DOS) giúp cho người sử dụng dễ dàng vận hành máy tính hơn khi sử dụng lệnh máy nhị phân khó nhớ. Tuy nhiên, khi chạy các lệnh ký tự này, phải thông dịch chúng thành các lệnh nhị phân nhờ bộ dịch có trong hệ điều hành (ví dụ như file command.com là bộ dịch của hệ điều hành DOS). Giữa các máy tính sử dụng các hệ điều hành khác nhau, thì tại mức 3 có nhiều điểm khác nhau hơn so với ở mức 1 và mức 2.

6. Mức 4 là mức ngôn ngữ assembly hay hợp ngữ. Như đã trình bày ở mức

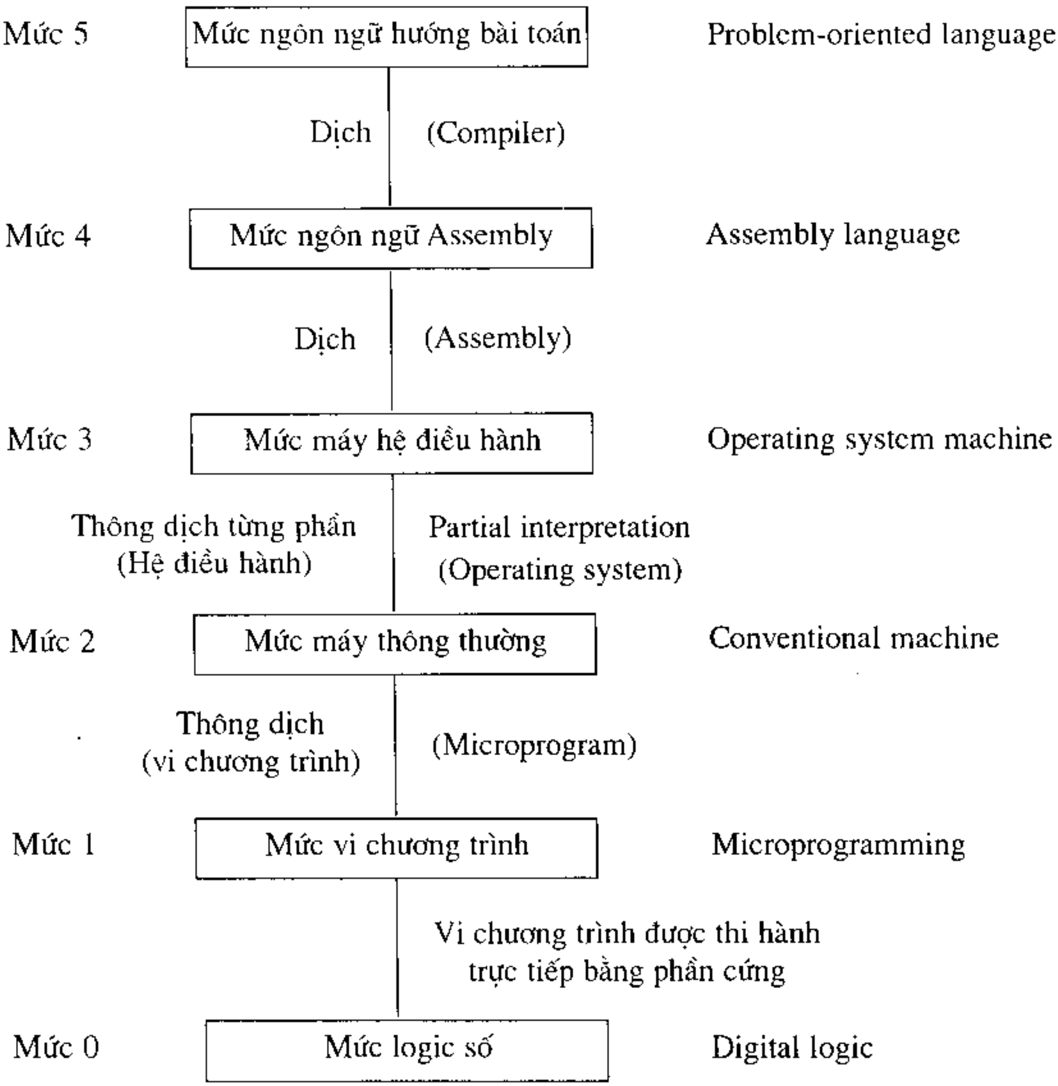
2, việc ra đời ngôn ngữ ký tự là cần thiết và làm cho người lập trình dễ nhớ các lệnh máy, công dụng của nó và nhờ vậy mà thuận lợi hơn trong lập trình. Đương nhiên khi đó, chương trình viết bằng ngôn ngữ ký tự phải được dịch ra lệnh máy nhị phân bằng một bộ dịch thì máy tính mới hiểu được. CPU chỉ có thể làm việc theo chương trình được viết trên ngôn ngữ máy. Chương trình đó gồm tập hợp các câu lệnh máy được mã hóa theo các số nhị phân 0 và 1. Khác với việc thông dịch (giải mã và chạy luôn) từ lệnh máy nhị phân ra vi chương trình, việc dịch ra lệnh máy nhị phân phải được thực hiện trước khi muốn chạy chương trình viết bằng ngôn ngữ ký tự. Việc dịch này tạo ra một bản dịch dưới dạng các lệnh máy nhị phân và bản dịch chạy được này được lưu vào bộ nhớ hay đĩa từ dưới dạng các tập tin (file) thường có đuôi .exe hay .com. Do việc tồn tại một bản dịch, người ta gọi cách dịch này là biên dịch.

Cần phải nhấn mạnh rằng, các mức 1, 2, 3 và một bên là mức 4 và mức 5 đề cập sau này, còn có sự khác nhau ở bản chất của các ngôn ngữ. Các ngôn ngữ máy của các mức 1, 2 và 3 là số. Bắt đầu từ mức 4, các ngôn ngữ gồm các từ (word) và những chữ viết tắt có ý nghĩa đối với con người. Mức 4 (và các mức bên trên nó) nhằm phục vụ những người lập trình ứng dụng để giải quyết các bài toán của họ. Mức 4 được gọi là mức ngôn ngữ assembly (dịch là ngôn ngữ lắp ghép và cũng còn được gọi là hợp ngữ), nó thực sự là một dạng ký hiệu (symbolic) cho một trong các ngôn ngữ ở bên dưới. Các ký hiệu này là các ký hiệu gợi nhớ (mnemonic), ví dụ như lệnh cộng được ký hiệu là Add, cũng vì nguyên nhân này mà mức 4 còn được gọi là ngôn ngữ gợi nhớ. Chương trình thực hiện công việc dịch chương trình viết bằng ngôn ngữ assembly sang một ngôn ngữ ở mức thấp hơn được gọi là assembler. Sở dĩ được gọi là assembler hay hợp ngữ vì từng dòng lệnh của ngôn ngữ này được dịch ra ngôn ngữ L2 hoặc L3 một cách tương ứng và ghép lại, tạo ra một bản dịch.

7. Mức 5 là mức ngôn ngữ bậc cao. Lập trình bằng hợp ngữ đòi hỏi lập trình viên phải hiểu hệ thống máy tính, Một công việc không dễ dàng và dễ chịu chút nào. Vì thế ngôn ngữ bậc cao ra đời. Các ngôn ngữ bậc cao được thiết kế cho những người lập trình ứng dụng sử dụng để giải các bài toán khác nhau của họ. Có hàng trăm ngôn ngữ bậc cao từng tồn tại, trong đó có một số ít được nhiều người biết đến là BASIC, C, FORTRAN VÀ PASCAL... Ngôn ngữ bậc cao có độ tích hợp lớn, mỗi dòng lệnh của ngôn ngữ bậc cao có thể dịch ra nhiều lệnh máy nhị phân. Vì thế ngôn ngữ bậc cao rất gần với con người, hay nói một cách khác, lập trình viên sử dụng ngôn ngữ bậc cao không nhất thiết phải nghiên cứu các mức dưới. Chương trình được viết bằng các ngôn ngữ này thường được dịch

sang mức 2 ,3 hoặc mức 4 bằng các trình dịch có trong ngôn ngữ đó, chúng được gọi là compiler (bộ biên dịch) và được viết bởi các nhà lập trình hệ thống hay những nhà sáng tạo ngôn ngữ đó.

Từ mức 2 đến mức 5 chính là phần mềm của máy tính. Như vậy có thể nói máy tính gồm từ mức –1 đến mức 1 là máy tính thực. Máy tính thực này cộng với ngôn ngữ ở các mức trên ta được máy tính ảo ở mức đó. Ví dụ, máy tính thực cộng với ngôn ngữ mức 2 ta được máy tính ảo mức 2. Sau đây chúng ta sẽ nghiên cứu kỹ lưỡng hơn các mức thiết bị, logic số, mức vi chương trình, mức hệ điều hành và hợp ngữ. Giản đồ sau mô tả các mức máy tính không kể mức –1.



II. MỨC THIẾT BỊ

Mức thiết bị như đã trình bày ở trên, nhiệm vụ chủ yếu là thiết kế các sơ đồ logic cơ bản - các viên gạch để xây dựng lên mức logic và các loại bộ nhớ.

1. Thiết kế các sơ đồ logic cơ bản

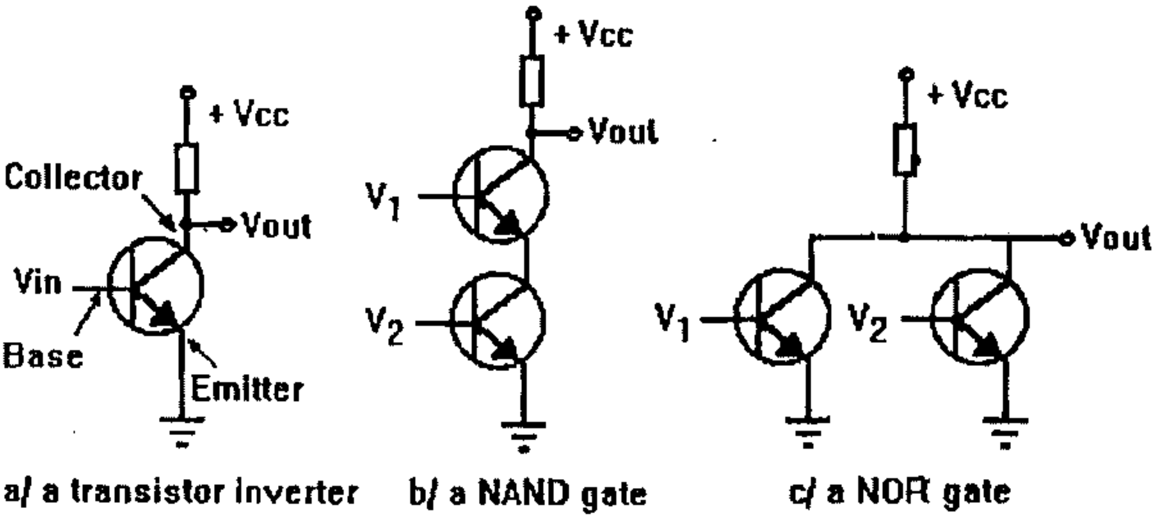
Mạch điện tử số là mạch điện trong đó chỉ biểu diễn 2 giá trị logic 0 và 1. Thí dụ tín hiệu điện nằm trong khoảng 0V – 0,8V (vôn) biểu diễn một giá trị, chẳng hạn 0 (sau này chúng ta có thể sẽ gọi là mức thấp hoặc low), còn tín hiệu điện nằm trong khoảng 3V – 5V biểu diễn giá trị kia, chẳng hạn 1 (sau này chúng ta có thể sẽ gọi là mức cao hoặc high). Các điện áp nằm ngoài miền này là không sử dụng. Những thiết bị điện tử rất nhỏ bé có thể thực hiện các chức năng khác nhau trên các giá trị nhị phân này được gọi là sơ đồ logic cơ bản hay cổng, chính các cổng tạo nên cơ sở phần cứng của tất cả các loại máy tính. Cổng có một hoặc một số lối vào (input) nhưng chỉ có một lối ra (output). Các giá trị vào hoặc ra chỉ có thể nhận một trong hai giá trị là 1 hoặc 0. Cổng được gọi là mạch logic bởi nó thực hiện các phép tính đại số logic.

Toàn bộ logic số hiện đại dựa trên thực tế là một transistor có thể được chế tạo để hoạt động như một chuyển mạch nhị phân hoạt động rất nhanh. Trên hình 3-1a là một transistor lưỡng cực được nối với một mạch điện đơn giản. Transistor này có 3 chân nối với bên ngoài, có tên là collector (cực góp), emitter (cực phát) và base (cực gốc).

Khi điện áp vào V_{in} thấp hơn một giá trị nào đó, transistor sẽ cấm (không có dòng điện đi qua) và thể hiện như một điện trở lớn vô cùng, làm cho lối ra của mạch điện là V_{out} nhận một giá trị xấp xỉ bằng V_{cc}, đó là một điện áp được ổn áp ở bên ngoài, thường là + 5V.

Khi điện áp vào V_{in} vượt quá một giá trị giới hạn nào đó (0,6V), transistor sẽ mở thông và thể hiện như một dây dẫn, làm cho lối ra của mạch điện là V_{out} được nối xuống đất, thường được quy ước là 0V.

Điều quan trọng là khi V_{in} thấp thì V_{out} cao và ngược lại. Mạch điện này chính là bộ đảo (a transistor inverter), đảo giá trị logic 0 thành 1 và 1 thành 0. Điện trở trong mạch điện dùng để hạn chế dòng điện đi qua transistor và tạo thành phân thế với transistor. Thời gian cần thiết để transistor chuyển từ trạng thái này sang trạng thái khác thường bằng vài nanô giây (1ns = 10^{-9}s).



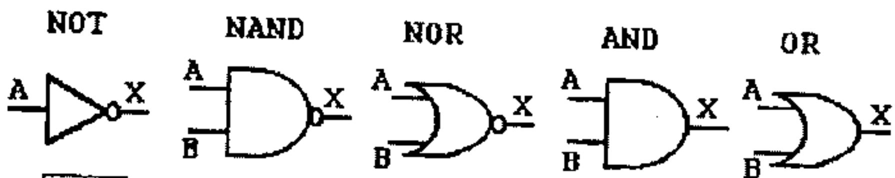
Hình 3-1

Trên hình 3-1b hai transistor được chồng nối tiếp lên nhau. Nếu cả V_1 và V_2 đều có mức cao (1) thì cả hai transistor đều dẫn điện và V_{out} sẽ được nối đất. Nếu một lối vào có mức thấp (0) thì transistor tương ứng sẽ cấm và V_{out} sẽ có mức cao. Nói tóm lại V_{out} có mức thấp khi và chỉ khi cả hai giá trị V_1 và V_2 là cao. Mạch điện này gọi là cổng và - đảo (a nand gate)

Trên hình 3-1c hai transistor được mắc song song. Theo cách mắc này nếu một lối vào nào đó có mức cao thì transistor tương ứng sẽ thông và lối ra được nối xuống đất. Nếu cả hai lối vào có mức thấp thì lối ra có mức cao. Mạch điện này gọi là cổng hoặc - đảo (a nor gate)

Cả 3 mạch điện trên hình 3-1 hoặc các mạch tương đương của chúng tạo nên các cổng đơn giản nhất. Chúng lần lượt được gọi là cổng NOT, cổng NAND và cổng NOR. Cổng NOT còn có tên gọi là Inverter (đảo). Nếu chúng ta quy ước rằng điện cao áp - high (V_{cc}) dùng để biểu diễn giá trị logic 1, còn điện áp thấp - low (đất) biểu diễn giá trị logic 0 thì chúng ta có thể biểu diễn giá trị ra như một hàm của các giá trị vào.

Các ký hiệu quy ước được sử dụng để mô tả các cổng này cùng với chức năng của chúng được biểu diễn trên hình 3-2a-c.



A	X
0	1
1	0

a/

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

b/

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

c/

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

d/

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

e/

Hình 3-2. Ký hiệu và hành trạng chức năng của 5 cổng cơ bản

Nếu tín hiệu ra của cổng NAND trên hình 3-2b lại được đưa tới đầu vào của một mạch đảo, thì chúng ta có một mạch mà đầu ra của nó bằng 1 khi và chỉ khi các đầu vào đồng thời bằng 1. Mạch điện đó được gọi là cổng AND, ký hiệu và mô tả chức năng (bảng chân lý) của nó được biểu diễn trên hình 3-2d.

Tương tự như trên, đầu ra của cổng NOR nếu nối tới đầu vào của một cổng NOT thì tạo thành một mạch điện mà giá trị đầu ra của nó sẽ bằng 0 khi và chỉ khi giá trị ở cả hai đầu vào bằng 0, còn ngược lại thì sẽ bằng 1. Mạch điện đó được gọi là cổng OR, ký hiệu và mô tả chức năng của nó được biểu diễn trên hình 3-2e.

Vòng tròn nhỏ ở đầu ra các cổng NOT, NAND, NOR được gọi là vòng tròn đảo, chúng được sử dụng cả trong các ngữ cảnh khác để chỉ một tín hiệu bị đảo.

Năm cổng trên hình 3-2 là những phần tử cơ bản xây dựng lên mức logic số (tuy nhiên theo lý thuyết hệ hàm đủ thì không cần đủ cả năm phần tử này). Từ những nghiên cứu trên ta thấy rằng các cổng NAND và NOR mỗi cổng chỉ cần sử dụng 2 transistor, trong khi đó các cổng AND và OR lại cần đến 3 transistor. Như vậy các cổng NAND và NOR đơn giản hơn các cổng AND và OR. Chính vì lý do này mà nhiều máy tính được xây dựng dựa trên các cổng NAND và NOR chứ không phải là các cổng AND và OR.

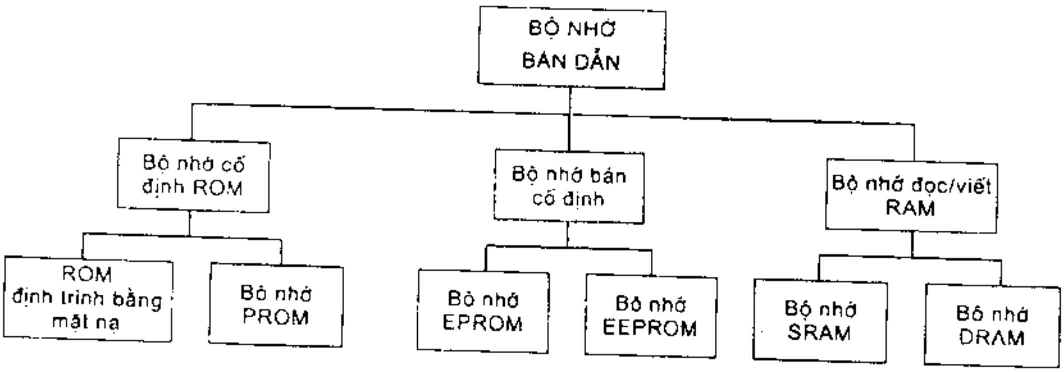
2. Thiết kế bộ nhớ

Bộ nhớ bán dẫn, nhờ tính tương thích về kích thước và mức lôgic, có tốc độ truy cập cao, năng lượng tiêu thụ thấp nên được thiết kế làm bộ nhớ trong (internal memory) nằm trên bản mạch chính cùng với CPU. Chúng được chế tạo như những chip riêng biệt hoặc được tích hợp trên cùng một chip với CPU. Xu thế ngày nay là phải tạo ra các chip nhớ vừa có dung lượng lớn với kích thước nhỏ vừa có tốc độ cao với giá thành thấp. Trong khi các chip nhớ thông thường có

kích thước cấu trúc điển hình cỡ 1μm thì các chip nhớ 64 Mbit giảm tới 0,3μm với 200 triệu transistor, tụ điện và điện trở phải cùng hoạt động chính xác với tốc độ cao trên cùng một chip. Chỉ cần một trong số các phần tử đó có vấn đề thì cả chip không thể hoạt động được. Do vậy việc chế tạo các bộ nhớ của PC hiện nay luôn phải đối mặt với các vấn đề thuộc về công nghệ cao (high technology).

2.1. Phân loại bộ nhớ bán dẫn

Hình 3-3 chỉ ra cách phân loại này.



Hình 3-3

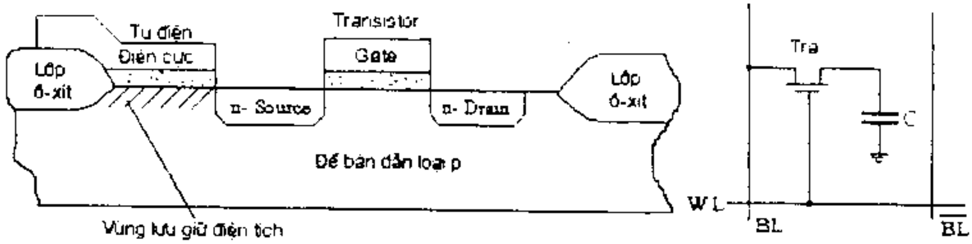
Bộ nhớ có nội dung được viết sẵn một lần khi chế tạo gọi là bộ nhớ cố định và được ký hiệu ROM (read only memory). Sau khi đã được viết (bằng mặt nạ) từ nhà máy thì ROM loại này không viết lại được nữa. PROM là một dạng khác, các bit có thể được viết bằng thiết bị ghi của người sử dụng trong một lần (programmable ROM). Bộ nhớ có thể đọc/viết nhiều lần gọi là RAM (random access memory) gồm có hai loại: bộ nhớ RAM tĩnh - SRAM (static RAM) thường được xây dựng trên các mạch điện tử flip - flop và RAM động - DRAM (dynamic RAM) được xây dựng trên cơ sở nhớ các điện tích ở tụ điện, bộ nhớ này phải được hồi phục nội dung đều đặn, nếu không nội dung sẽ mất theo sự rò điện tích trên tụ. Giữa ROM và RAM có một lớp các bộ nhớ được gọi là EPROM (erasable programmable ROM), dữ liệu trong đó có thể xóa được bằng tia cực tím và ghi lại được, EEPROM (electric earasable PROM) có thể xóa được bằng dòng điện. Các loại này còn được gọi là bộ nhớ bán cố định. Các bộ nhớ DRAM thường thỏa mãn những yêu cầu khi cần bộ nhớ có dung lượng lớn, trong khi đó khi cần có tốc độ truy xuất lớn thì lại phải dùng các bộ nhớ SRAM với giá thành đắt hơn. Nhưng cả hai loại này đều có nhược điểm là thuộc loại

"bay hơi" (volatile), thông tin sẽ bị mất đi khi nguồn điện nuôi bị cắt. Do vậy các chương trình dùng cho việc khởi động PC như BIOS thường phải nạp trên các bộ nhớ ROM hoặc bộ nhớ bán cố định.

2.2. Bộ nhớ đọc viết

a. DRAM

Một ô nhớ của DRAM như hình 3-4 gồm có một transistor trường MOS có trở lối vào rất lớn và một tụ điện C là linh kiện lưu giữ một bit thông tin tương ứng với hai trạng thái có hoặc không có điện tích trên tụ.



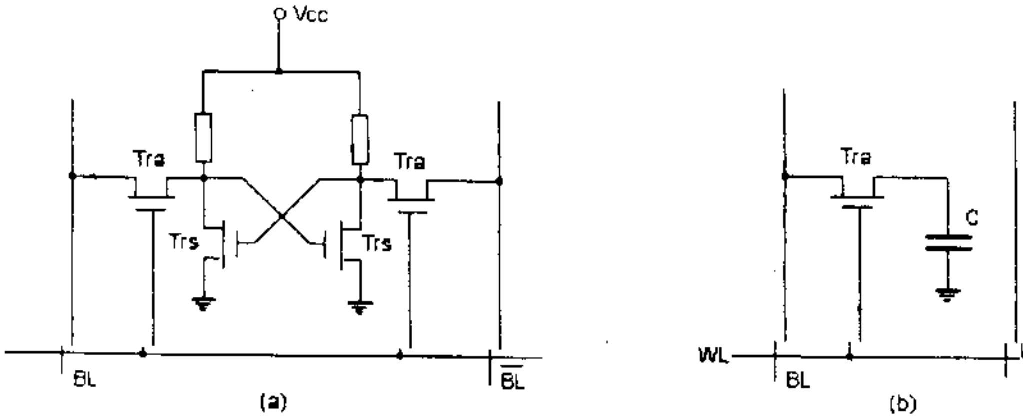
Hình 3-4

Transistor hoạt động như một công tắc, cho phép nạp hay phóng điện tích của tụ khi thực hiện phép đọc hay viết. Khi đọc, viết hàng nào, các transistor trên hàng đó sẽ được mở bởi tín hiệu đọc/viết. Có thể hiểu một cách đơn giản, khi có một tổ hợp nhị phân phát trên bus A, tác động vào đầu vào của bộ giải mã địa chỉ thì bộ giải mã địa chỉ sẽ phát tín hiệu đọc/viết vào hàng nhớ có chỉ số hàng bằng giá trị thập phân của tổ hợp nhị phân đó và làm mở các transistor của hàng đó. Cực cửa (gate) của transistor được nối với dây hàng (còn gọi là dây từ WL) và cực máng (drain) được nối với dây cột (còn được gọi là dây bit BL hoặc BL đảo), cực nguồn (source) được nối với tụ điện. Khi transistor mở thì tụ điện nhớ sẽ được nối vào dây bit thông qua bộ khuếch đại. Điện áp nạp trên tụ tương đối nhỏ nên một bộ khuếch đại nhạy được sử dụng trong bộ nhớ. Khi vi xử lý đọc bộ nhớ, bộ khuếch đại sẽ khuếch đại mức điện áp này lên 5V truyền vào bus D. Khi viết thông tin vào bộ nhớ, transistor cũng mở như khi đọc, điện áp 5V sẽ được nạp cho tụ điện không thông qua khuếch đại. Như vậy có thể coi như có một công tắc mắc song song với bộ khuếch đại và khi đọc thì công tắc này ngắt, còn khi viết thì công tắc này đóng làm ngắn mạch bộ khuếch đại, do vậy không còn hiệu ứng khuếch đại nữa. Trong một sơ đồ tương đương như vậy, công tắc được điều khiển bởi tín hiệu đọc/viết (R/W). Do dòng rò của transistor,

ô nhớ cần được nạp lại trước khi điện áp trên tụ thấp hơn một ngưỡng nào đó. Quá trình này được thực hiện nhờ một chu kỳ "làm tươi" (refresh), khi đó điện áp trên tụ được xác định (ở trạng thái 0 hay 1) và mức điện áp logic này được khuếch đại lên thành 5V khi có lệnh đọc giả và viết lại vào ô nhớ bằng lệnh viết giả.

b. SRAM

Một ô nhớ của SRAM giữ thông tin bởi trạng thái của mạch triger như hình



Hình 3-5

3-5a khi so sánh với ô nhớ DRAM như hình 3-5b.

Giả sử Trs trái đóng, thế trên cực máng sẽ là V_{cc} (thường là bằng 5V), thế này tác động vào cực cửa của Trs phải làm cho nó mở bão hoà và thế trên cực máng bằng 0V và chính thế này tác động vào cực cửa của Trs trái và ghim Trs trái ở trạng thái đóng một cách bền vững và vì vậy trên cực máng của Trs trái luôn lưu giữ mức logic 1. Thuật ngữ "tĩnh" chỉ ra rằng khi nguồn nuôi chưa bị cắt thì nội dung của ô nhớ vẫn được giữ nguyên và không cần quá trình làm tươi bộ nhớ. Khác với ô nhớ DRAM, ở đây ô nhớ triger cung cấp một tín hiệu số liệu mạnh hơn nhiều vì đã có các transistor trong các ô nhớ, chúng có khả năng khuếch đại tín hiệu (chế độ khoá) và do đó có thể cấp trực tiếp cho các đường bit các thế chuẩn. Trong DRAM, sự khuếch đại tín hiệu trong các bộ khuếch đại cần nhiều thời gian và do đó thời gian thâm nhập dài hơn. Khi định địa chỉ các triger ở SRAM, các transistor bổ sung cho các ô triger, các bộ giải mã địa chỉ... cũng được đòi hỏi như ở DRAM.

Như trong DRAM, cực cửa của transistor ở đây cũng được nối với đường từ

và cực máng được nối với cặp đường bit. Nếu số liệu được đọc từ ô nhớ, khi ấy bộ giải mã hàng kích hoạt dây từ WL tương ứng. Hai transistor Tra được thông và nối hai cực máng của triger nhớ với cặp dây bit.

Viết số liệu được thực hiện theo cách ngược lại, bộ giải mã hàng kích hoạt đường dây từ và làm thông transistor Tra, nối hai cực máng của triger nhớ với cặp dây bit. Mức điện thế trên dây bit mà trùng với mức điện thế trên cực máng một cách tương ứng thì thông tin được giữ nguyên. Nếu không trùng thì các cực máng sẽ lấy giá trị điện thế của các dây bit, vì điện thế trên dây bit xuất hiện trong thời gian viết số liệu có tác dụng như xung đảo triger. Do đó triger sẽ chuyển trạng thái phù hợp với số liệu mới hoặc giữ giá trị đã được lưu giữ phụ thuộc vào việc số liệu viết trùng với số liệu đã lưu giữ hay không.

Một thí dụ là chip SRAM của hãng Intel-51258 có dung lượng nhớ 256 Kbit với tổ chức 64Kx4bit, do đó cần 16 đầu địa chỉ từ A_0 đến A_{15}.

2.3. Bộ nhớ cố định

a. ROM

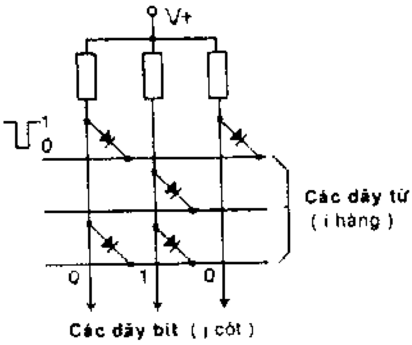
Các chip RAM không thích hợp cho các chương trình khởi động do các thông tin trên đó bị mất khi tắt máy PC. Do vậy phải dùng đến ROM, trong đó các số liệu cần lưu trữ được viết vào một lần theo cách không bay hơi (non-volatile) để nhằm lưu giữ được lâu dài.

ROM lập trình kiểu mặt nạ được gọi đơn giản là ROM. Nó được chế tạo trên một phiến si-lic theo một số bước xử lý như quang khắc và khuếch tán để tạo ra những tiếp giáp bán dẫn có tính dẫn điện theo một chiều (như diode, transistor trường). Người thiết kế định rõ chương trình muốn ghi vào ROM và thông tin này được sử dụng để điều khiển quá trình làm mặt nạ.

Hình 3-6 là một thí dụ đơn giản về một sơ đồ ROM dùng diode.

Mỗi vị trí vắt chéo nhau giữa các dây từ (hàng) và các dây bit (cột) tạo nên một phần tử nhớ (ô nhớ 1 bit). Một diode được đặt tại đó như hình vẽ sẽ cho phép lưu trữ số liệu "0". Ngược lại, những vị trí vắt chéo không có diode sẽ cho phép lưu trữ số liệu "1". Khi đọc một từ số liệu thứ i của ROM, bộ giải mã sẽ đặt dây từ đó xuống mức lôgic thấp, các dây còn lại ở mức cao. Do vậy chỉ những diode nối với dây này được phân cực thuận trở nên thông làm cho thế lối ra trên các dây bit tương ứng ở mức thấp (mức logic "0"). Các dây bit còn lại sẽ giữ ở mức cao (mức logic "1").

Cả hai công nghệ MOS và lưỡng cực dùng để chế tạo ROM. Thời gian thâm nhập của bộ nhớ lưỡng cực khoảng từ 50 - 90ns còn ở bộ nhớ MOS dài hơn 10 lần.



Hình 3-6

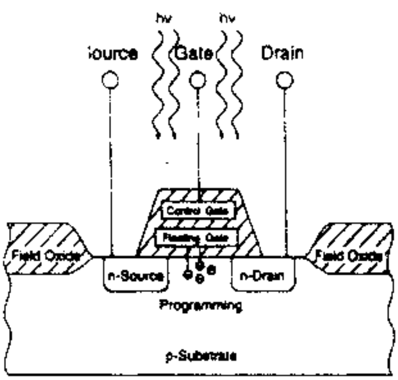
b. PROM

Nó cũng gồm các diode như trên chúng có mặt đầy đủ ở tất cả các vị trí vắt chéo. Mỗi diode lại được nối tiếp với một cầu chì. Bình thường khi chưa lập trình, các cầu chì còn nguyên vẹn, nội dung của PROM sẽ toàn là 0. Khi định vị đến một bit bằng cách đặt một xung điện ở lối ra tương ứng, cầu chì sẽ bị đứt và bit này sẽ là 1. Bằng cách đó ta có thể lập trình toàn bộ các bit trong PROM. Như vậy, lập trình có thể được thực hiện bởi người sử dụng chỉ một lần duy nhất, không thể sửa đổi lại được.

2.4. Bộ nhớ bán cố định

a. EPROM (Erasable PROM)

Số liệu có thể được viết vào bằng điện nhưng được lưu giữ theo kiểu không bay hơi (non-volatile). Đó là một ROM có thể lập trình và xóa được. Hình 3-7 chỉ ra cấu trúc của một transistor dùng để làm một ô nhớ gọi là FAMOST (Floating gate avalanche injection MOS transistor).



Hình 3-7

Trong ô nhớ dùng transistor này, cực cửa được nối với đường từ, cực máng nối với đường bit và cực nguồn nối với nguồn chuẩn (5V) được coi là nguồn cho mức lôgic 1. Khác với transistor MOS bình thường, ở đây có thêm một cửa gọi là cửa nổi (floating gate), đó là một vùng vật liệu được thêm vào giữa lớp cách điện cao. Nếu cửa nổi không có điện tích thì nó không ảnh hưởng gì tới cực cửa điều khiển và transistor hoạt động như bình thường. Tức là khi dây từ được kích hoạt (cực cửa có điện thế dương) thì transistor thông, cực máng và nguồn được nối với nhau qua kênh dẫn và dây bit có mức logic 1. Nếu cửa nổi có các điện tử trong đó có điện tích âm, chúng sẽ ngăn trường điện điều khiển của cực cửa điều khiển và dù dây từ đó được kích hoạt thì cũng không thể phát ra trường đủ mạnh với cực cửa điều khiển để làm thông transistor. Lúc này đường bit không được nối với nguồn chuẩn và ô nhớ coi như được giữ giá trị logic 0.

Việc nạp các điện tử vào vùng cửa nổi, tức là tạo ra các ô nhớ có giá trị lôgic 0, được thực hiện bởi các xung điện có độ dài cỡ 50ms và độ lớn +20V đặt vào giữa cực cửa và máng. Lúc đó những điện tử mang có năng lượng lớn sẽ đi qua lớp cách điện giữa đế (substrate) và cửa nổi. Chúng tích tụ trong vùng cửa nổi và được giữ ở đây sau khi xung chương trình tắt. Đó là do cửa nổi được cách điện cao với xung quanh và các điện tử không còn đủ năng lượng sau khi lạnh đi, để có thể vượt qua ngoài lớp cách điện đó nữa. Chúng sẽ được giữ ở đây trong một thời gian rất dài (ít nhất là 10 năm).

Để xóa các thông tin, tức là làm mất các điện tử trong vùng cửa nổi, phải chiếu ánh sáng tử ngoại UV vào chip nhớ. Những điện tử ở đây lúc này hấp thụ năng lượng nhảy lên các mức năng lượng cao, chúng sẽ rời cửa nổi như cách thâm nhập vào đó. Trong chip EPROM có một cửa sổ làm bằng thủy tinh thạch anh chỉ để cho ánh sáng tử ngoại đi qua khi cần xóa số liệu trong bộ nhớ.

b. EEPROM (Electrically Erasable PROM)

Cửa sổ thạch anh thì đắt và việc sử dụng nguồn phát tia tử ngoại không tiện lợi, nên những năm gần đây xuất hiện các chip PROM không đắt lắm có thể xóa dữ liệu bằng phương pháp điện. Việc nạp các điện tử cho cửa nổi được thực hiện như cách ở EPROM. Bằng một xung điện thế tương đối dài, các điện tích mang năng lượng cao từ trong đế sẽ thấm qua lớp cửa và tích tụ trong cửa nổi. EEP-ROM có cấu tạo sao cho cửa nổi có một phần sát xuống dưới đế và cực máng tạo thành một kênh (tunnel) màng mỏng ôxit. Các lớp cách điện không thể là lí

tưởng được, các điện tích mang có thể thấm qua lớp phân cách với một xác suất thấp. Xác suất này tăng lên khi bề dày của lớp giảm đi và điện thế giữa hai điện cực ở hai mặt lớp cách điện tăng lên. Muốn phóng các điện tích trong vùng cửa nổi, một điện thế âm 20V được đặt vào cực cửa điều khiển và cực máng. Lúc này các điện tử âm trong cửa nổi được chảy về cực máng qua kênh màng mỏng ôxit và số liệu lưu giữ được xóa đi. Điều lưu ý là phải làm sao cho dòng điện tích này chảy qua không quá lâu, vì nếu không vùng cửa nổi này lại trở nên tích điện dương làm cho hoạt động của transistor không được ở trạng thái bình thường.

Các chip ROM hiện nay có thời gian thâm nhập cỡ từ 120ns đến 150ns dài hơn nhiều thời gian đó trong các chip nhớ RAM.

Các ô nhớ miêu tả trên là các ô nhớ 1bit, các ô nhớ 8 bit thường được sử dụng do 8 ô nhớ 1 bit ghép lại theo hàng. Từ các ô nhớ này sẽ tạo lên bộ nhớ dung lượng lớn hàng triệu hàng.

III. MỨC LOGIC SỐ

Mức sát dưới cùng của kiến trúc phân tầng của một máy tính là mức logic số, đó là phần cứng thực sự của máy tính. Phần này nghiên cứu một số các mạch điện xây dựng nên mọi máy tính số và đại số logic dùng để phân tích chúng.

1. Đại số logic

Các mạch điện số có thể được xây dựng từ một số ít các phần tử rất đơn giản là các sơ đồ lôgic cơ bản. Để xây dựng các mạch điện tổ hợp thực hiện một chức năng, một hàm nào đó ví dụ như bộ cộng, bộ nhân... từ các cổng - các phần tử lôgic cơ bản nói trên cần sử dụng đại số logic hay đại số Boole. Trong đại số logic, cả biến và hàm logic đều thuộc tập hợp chỉ có hai giá trị 0 và 1. Quan hệ phụ thuộc của hàm vào n biến có thể biểu diễn thông qua bảng chân lý như đối với hàm And, Nand trình bày ở phần trên. Tuy nhiên, cách biểu diễn này khá cồng kềnh. Người ta thường dựa vào định lý về chuẩn tắc tuyển (tổng) để biểu diễn hàm dưới dạng đại số:

"Một hàm logic n biến bất kỳ luôn có thể biểu diễn dưới dạng chuẩn tắc tổng đầy đủ tức là tổng của nhiều thành phần, mỗi thành phần là một tích gồm n biến. Có bao nhiêu lần hàm bằng 1 thì có bấy nhiêu tích. Trong mỗi tích biến bằng 1 giữ nguyên, còn biến bằng 0 lấy phủ định".

Tương tự như vậy, có định lý về chuẩn tắc hội:

" Một hàm logic n biến bất kỳ luôn có thể biểu diễn dưới dạng chuẩn tắc hội đầy đủ tức là tích của nhiều thành phần, mỗi thành phần là một tổng gồm n biến. Có bao nhiêu lần hàm bằng 0 thì có bấy nhiêu tổng. Trong mỗi tổng biến bằng 0 giữ nguyên, còn biến bằng 1 lấy phủ định"

Ví dụ hàm And của hai biến vào A và B, khi đó được viết dưới dạng chuẩn tắc tổng: f=A.B. Còn dạng chuẩn tắc hội như sau: f=(A+B).(/A+B).(A+/B). Hai dạng này hoàn toàn tương đương nhau, nhưng dạng chuẩn tắc tổng đơn giản hơn.

Đến đây, qui trình thiết kế mạch đã rất sáng sủa, ví dụ với hàm chuẩn tắc tổng:

- Xác định hàm có bao nhiêu lần bằng 1 (có thể thông qua bảng chân lý) từ đó viết ra bấy nhiêu tích của đủ n biến và sử dụng sơ đồ And để thực hiện tích đó. Các biến nối vào sơ đồ And nếu bằng 1 thì nối thẳng, còn nếu bằng 0 thì nối qua sơ đồ đảo.

- Đầu ra của các sơ đồ And đấu vào sơ đồ Or. Đầu ra của sơ đồ Or chính là hàm logic cần tìm, cần thiết kế.

Tiếp theo, người ta thấy rằng có thể tiết kiệm các phần tử logic cơ bản bằng cách rút gọn hàm đại số lôgic (tối thiểu hoá) như trong đại số thường trước khi thiết kế mạch, dựa vào các hệ thức đặc biệt hay phương pháp bìa Cacnô.

Một vấn đề quan trọng cần nêu ra ở đây là hệ hàm đủ: mọi hàm logic có thể biểu diễn thông qua các hàm logic cơ bản: And, Or, Not. Tức là, mọi hàm logic có thể được thiết kế chỉ bằng các sơ đồ logic cơ bản And, Or, Not.

Dựa vào các hệ thức đặc biệt và các định lý, người ta dễ dàng chứng minh được 4 hệ hàm: [And,Not], [Or,Not], [Nand], [Nor] cũng là những hệ hàm đủ tương đương.

2. Các mạch logic số cơ bản

Ngày nay rất hiếm vi mạch được chế tạo ra mà chỉ có một cổng, các vi mạch có bán trên thị trường thường chứa một số cổng để giảm giá thành, kích thước...

2.1. Mạch tích hợp (integrated circuits - IC)

Các cổng được sản xuất ra không phải dưới dạng mỗi đơn vị sản phẩm chứa một cổng mà trong một đơn vị có chứa đựng nhiều cổng, người ta gọi những đơn vị này là mạch tích hợp, hay còn gọi là IC (integrated circuits) hay là chip. Các chip thường có kích thước trong khoảng rộng cỡ (5-15)mm, dài cỡ (20-50)mm. Số chân của một chip có thể là 14, 16, 18, 20... 68... Các chip lớn có

chân chìa ra ở cả 4 cạnh.

Người ta thường phân loại chip theo số lượng cổng chứa trong chip một cách tương đối thành một số loại như trong bảng sau.

Ký hiệu	Số cổng/chíp
SSL (small scale integrated)	1-10
MSI (medium scale interated)	10-100
LSI (large scale interated)	100-100.000
VLSI (very large scale interated)	> 100.000

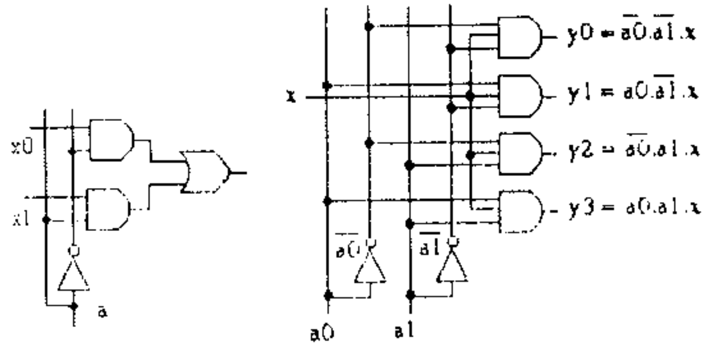
Hiện nay người ta có thể đặt hàng triệu cổng trong một chip, chip có một triệu cổng có thể cần tới 3.000.000 chân (1 triệu cổng x 3 chân/cổng + 2chân cho nguồn nuôi), tuy nhiên điều này không thể chấp nhận được. Trong thực tế các chip có nhiều cổng thường chứa một số đơn vị logic để thực hiện các chức năng nhất định, mỗi đơn vị logic thường được xây dựng từ một số cổng, các cổng này liên hệ với nhau thông qua các đường dây dẫn ngay bên trong chip, có thể chỉ cần một số ít, thậm chí không cần các chân đưa ra ngoài. Các đơn vị chức năng trong một chip cũng có thể liên hệ với nhau bằng các đường dây dẫn bên trong chip. Một số chân của chip có thể được các đơn vị chức năng dùng chung theo kiểu phân chia thời gian. Trong công nghệ chế tạo chip người ta phải thiết kế các chip sao cho có tỉ số cổng/chân ra cao.

2.2. Các mạch tổ hợp (combinational circuit)

Nhiều ứng dụng logic số cần đến các mạch có nhiều đầu vào và nhiều đầu ra trong đó các giá trị hoàn toàn được xác định bởi các giá trị đầu vào ở thời điểm đang xét. Những mạch như vậy được gọi là mạch tổ hợp (combinational circuit), các mạch này thường được thể hiện bởi bảng chân lý. Không phải tất cả các mạch điện đều có tính chất này (chẳng hạn các mạch nhớ). Trong mục này sẽ khảo sát một số mạch tổ hợp thông dụng, chúng thường là các chip MSI. Đặc điểm quan trọng của các mạch này là hoạt động của nó đã được điều khiển bằng từ điều khiển.

a. Bộ dồn kênh (multiplexer)

Là một mạch điện với 2^n lối vào số liệu, một lối ra số liệu, và n lối vào điều khiển. Một trong số 2^n đầu vào số liệu sẽ được chọn để xuất ra lối ra, tuỳ theo giá trị thập phân của tín hiệu điều khiển. Ví dụ nếu giá trị này bằng k, thì dữ liệu ở đầu vào thứ k được xuất ra lối ra. Hình 3-8a là một dồn kênh (mux) có 2 đường vào số liệu (x_0,x_1), 1đường vào điều khiển (a). Mux có một ứng dụng quan trọng trong thiết kế bộ số học lôgic ALU và hoạt động như một bộ chuyển mạch điện tử.



a/ Multiplexer b/ Demultiplexer

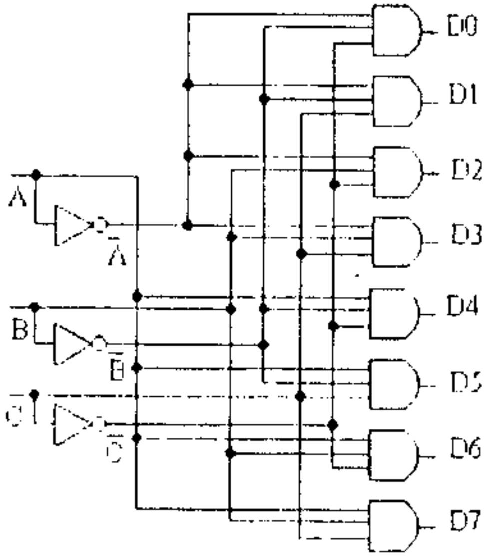
Hình 3-8

b. Bộ phân kênh (de multiplexer)

Mạch điện thực hiện chức năng ngược với mux được gọi là demultiplexer (demux), nó cho tín hiệu vào (chỉ có một đầu vào) đi ra một trong số 2^n đường ra tùy theo tín hiệu trên n đầu vào điều khiển. Nếu giá trị thập phân của tín hiệu điều khiển bằng k thì đường ra thứ k được chọn để đi ra. Hình 3-8b là một demux có 1 đường vào số liệu x, 2 đường vào điều khiển a_0, a_1 và 4 đường ra y_0, y_1, y_2 và y_3. Một ứng dụng quan trọng của demux là chế tạo bộ giải mã hàng bộ nhớ.

c. Bộ giải mã (decoder).

Có nhiều loại bộ giải mã. Bộ giải mã dùng nhiều trong kỹ thuật máy tính là loại giải mã 1 từ n. Đó là mạch điện có n lối vào và 2n lối ra. Khi tác động một tổ hợp nhị phân n bit vào n lối vào một cách tương ứng, thì lối ra duy nhất có số thứ tự bằng giá trị thập phân của tổ hợp nhị phân đó sẽ ở mức tích cực (giả sử bằng 1). Các lối ra còn lại ở mức không tích cực (theo giả sử trên phải bằng 0).



Hình 3-9

Hình 3-9 là một bộ giải mã với n = 3. Để thấy được công dụng của bộ giải mã, ta hình dung một bộ nhớ có 8 chip nhớ, mỗi chip 2KB, chip 0 có các địa chỉ từ 0 đến 2047, chip 1 có các địa chỉ từ 2048 đến 4095...

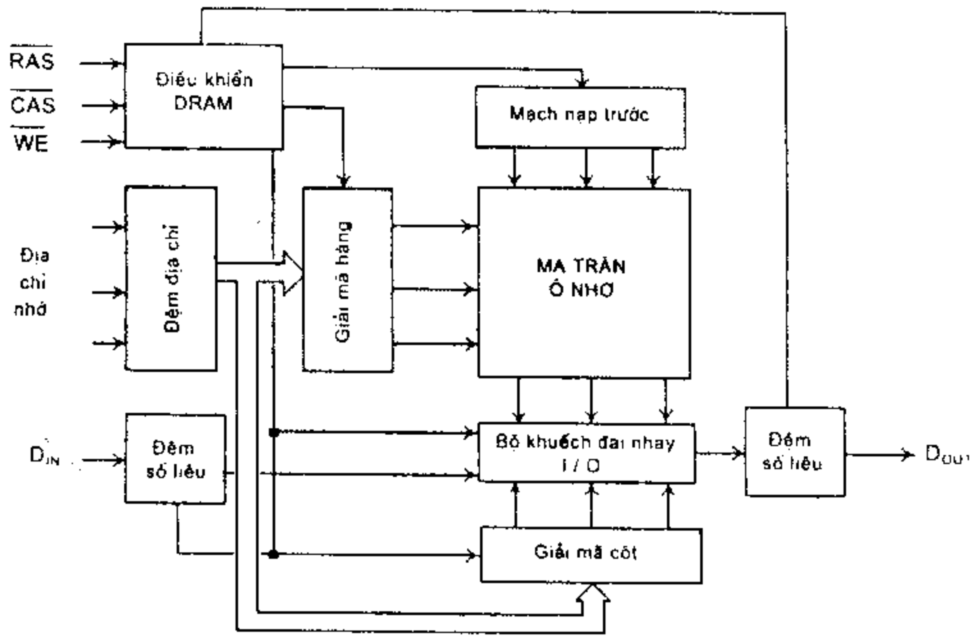
Khi bộ vi xử lý phát tín hiệu trên bus địa chỉ tới bộ nhớ, 3 bit bậc cao của bus A sẽ được dùng để chọn 1 trong 8 chip, đó là 3 bit A, B, C ở hình 3-9. Với 1 giá trị của con số 3 bit A, B, C chỉ có 1 dây Di (i = 0..7) có mức 1, các dây còn lại đều có mức 0, các dây này được nối vào đầu vào chọn chip (chipselect: CS hoặc CE) của 8 chip, vì vậy chỉ có 1 chip có CS =1 là được phép mở. Sau khi đã chọn xong chip, phải chọn, phải mở một ô nhớ 1B trong số 2KB của chip đã chọn để trao đổi dữ liệu với vi xử lý. Để giảm kích thước bộ giải mã, tức là giảm kích thước và giá thành vi mạch, đến đây người ta thường dùng giải mã hai bước như sau.

Sơ đồ tổ chức của một chip nhớ DRAM như hình 3-10. Các ô nhớ được sắp xếp theo hàng và cột trong một ma trận nhớ. Địa chỉ ô nhớ được chia thành hai phần, địa chỉ hàng và cột. Hai địa chỉ này được đọc vào bộ đệm một cách lần lượt. Quá trình giải mã này được điều khiển bởi các tín hiệu RAS (row access strobe) và CAS (column access strobe).

Bước 1, khi RAS ở mức tích cực thấp thì DRAM nhận địa chỉ được đặt vào nó và sử dụng như là địa chỉ hàng. Nếu một chip nhớ 2KB được chia thành 2^7 = 128 đoạn nhớ, thì một bộ giải mã 7 lối vào (nối với 7 đường địa chỉ của

busA) sẽ cung cấp 128 đường địa chỉ hàng, nhưng chỉ có duy nhất một đường là ở mức tích cực và mở đoạn nhớ tương ứng. Khi đó một đoạn nhớ trong bộ nhớ 2KB gồm 16 từ nhớ 8 bit sẵn sàng trao đổi tin.

Bước 2, khi CAS ở mức tích cực thấp thì DRAM nhận địa chỉ được đặt vào nó và sử dụng như là địa chỉ cột. Lúc này bộ giải mã cột sẽ chọn 1 trong 16 từ nhớ 8 bit đó. Có thể sử dụng 8 bộ dồn kênh 16 - 1 để làm bộ giải mã cột, trong đó 16 đầu vào của bộ dồn kênh thứ i sẽ được nối với 16 đường 2^i. Bốn đường địa chỉ của bus A được nối vào bốn lối vào điều khiển của tất cả các bộ dồn kênh một cách song song và tuỳ theo giá trị nhị phân trên 4 đường bus A nói trên, lối ra của 8 bộ dồn kênh sẽ được nối vào từng bit tương ứng của byte cần tìm. Tổng cộng lại, cần 14 đường địa chỉ nhớ để có thông tin giải mã chọn chip, chọn hàng và cột.

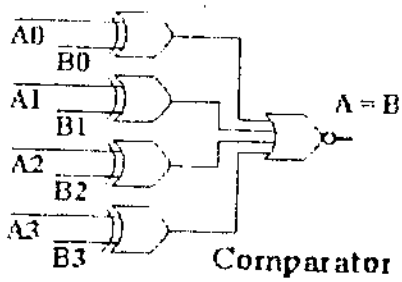


Hình 3-10

Bộ điều khiển nhớ của CPU sẽ phải thực hiện ba việc khác nhau: chia địa chỉ từ CPU thành các địa chỉ hàng và cột và cấp chúng cho DRAM một cách lần lượt, kích hoạt các tín hiệu RAS, CAS và WE một cách chính xác và truyền cũng như chấp nhận các số liệu đọc, viết.

d. Bộ so sánh (comparator)

Đó là mạch điện thực hiện so sánh 2 từ đưa vào, trên hình 3-11 là một bộ so sánh 2 toán hạng nhị phân 4 bit. Hai từ đưa vào là $A=A_3A_2A_1A_0$ và $B = B_3B_2B_1B_0$, trong đó A_i và B_i (i=0..3) là các bit bậc i của từ A và B. Đầu ra A=B sẽ có giá trị 1 nếu 2 từ đưa vào là bằng nhau, ngược lại sẽ có giá trị 0. Bộ so sánh rất tiện dụng để làm bộ giải mã địa chỉ cho thiết bị ngoại vi



Hình 3-11

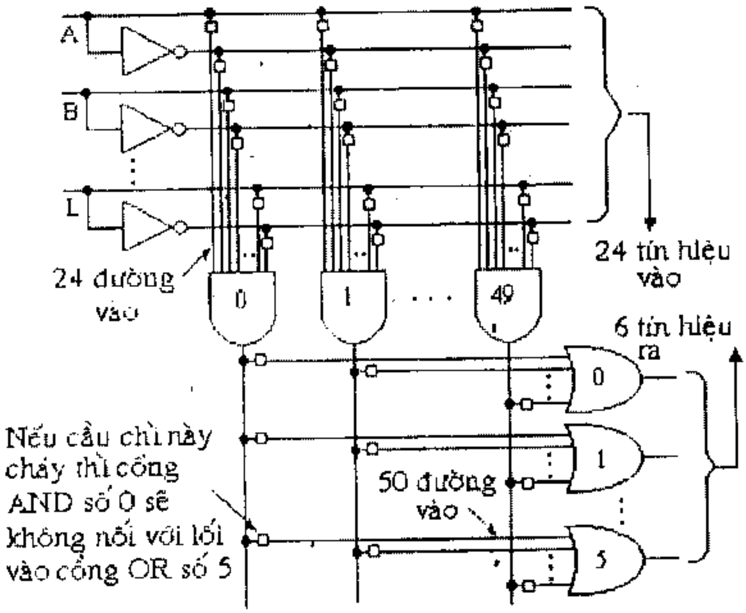
Mạch điện này dùng các cổng XOR 2 lối vào và 1 cổng NOR 4 lối vào. Trên thị trường cũng còn có các bộ so sánh với đầu ra A>B hoặc A<B.

e. Mảng logic định trình được - PLA (Programmed Logic Arrays)

Theo đại số logic, một hàm tuỳ ý hay một bảng chân lý có thể được xây dựng bởi việc lấy tích các số hạng bằng các cổng AND, sau đó cộng (OR) các số hạng ấy lại. Những chip điển hình để thực hiện cộng các tích lại là chip PLA.

Hình 3-12 là chíp PLA có ký hiệu 74S330, chip này có các đường vào cho 12 tín hiệu vào - biến, giá trị đảo cực của các biến vào được tạo thành bên trong chíp, như vậy có 24 biến vào.

Phần chính của mạch này là một ma trận 50 cổng AND, mỗi cổng này có thể nối các đầu vào của nó với một tập con tuỳ ý của 24 đường tín hiệu vào. Tín hiệu vào nào đi tới cổng AND nào được quyết định bởi một ma trận 24x50 bít do người sử dụng thiết lập. Mỗi tín hiệu vào trong số 24 tín hiệu đi tới đầu vào của tất cả các cổng phải qua 1 cầu chì. Ban đầu, khi mới xuất xưởng tất cả các cầu chì còn thông. Người sử dụng sẽ làm cháy các cầu chì được lựa chọn bằng cách sử dụng một thiết bị chuyên dụng.



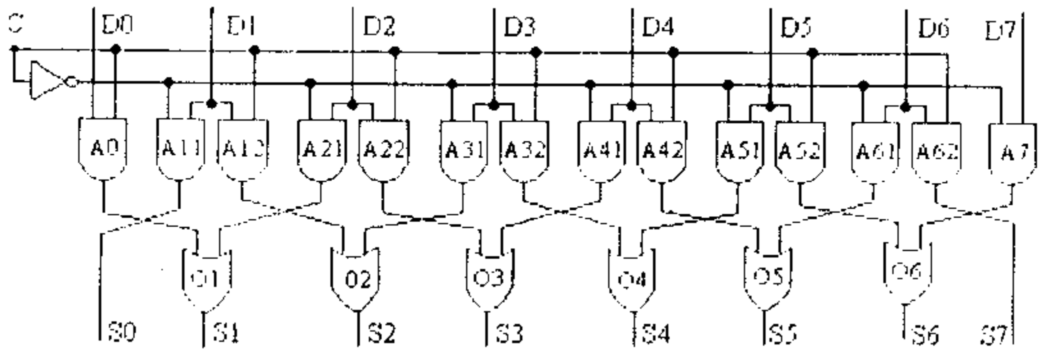
Hình 3-12

Phần ra của mạch điện có 6 cổng OR, mỗi cổng này có đến 50 đầu vào ứng với 50 đầu ra của 50 cổng AND, ở phần này ta cũng thấy một ma trận 50x6 do người sử dụng thiết lập các mối nối dây hàng - dây cột thông qua các cầu chì.

Chíp 74S330 có 12 chân cho 12 biến vào, 6 chân cho 6 đường dây ra, 1 chân cho nguồn nuôi, 1 chân cho dây đất, tổng cộng là 20 chân ra.

f. Bộ ghi dịch (shifter)

Dịch bit là một thao tác cơ sở, một số phép toán số học và logic của các bộ vi xử lý trong máy tính dựa trên các thao tác này. Trên hình 3-13 là bộ dịch bit có 8 đầu vào (D_0 ..... D_7), 8 đầu ra (S_0 .... S_7), đường điều khiển C xác định hướng dịch chuyển của các bit, nếu C = 1 thì dịch phải một vị trí, nếu C = 0 thì dịch trái 1 vị trí. Khi C = 1 các cổng AND bên trái trong các cặp cổng AND, như A_{11}, A_{21},.. A_{61} và A_7 đều có 1 lối vào bằng 0, do đó đầu ra của các cổng AND này đều bằng 0 và dẫn đến việc tất cả các cổng OR từ 01 đến 06 đều có lối vào bên phải bằng 0, vì vậy các đầu ra S_1... S_6 sẽ nhận giá trị của các đầu vào bên trái. Kết quả là $S_1 = D_0$, $S_2 = D_1$,.... và $S_6 = D_5$, còn lại $S_0 = 0$ và $S_7 = D_6$. Như vậy khi C=1 các bit đưa vào đã được dịch phải một vị trí và đưa ra lối ra.



Hình 3-13

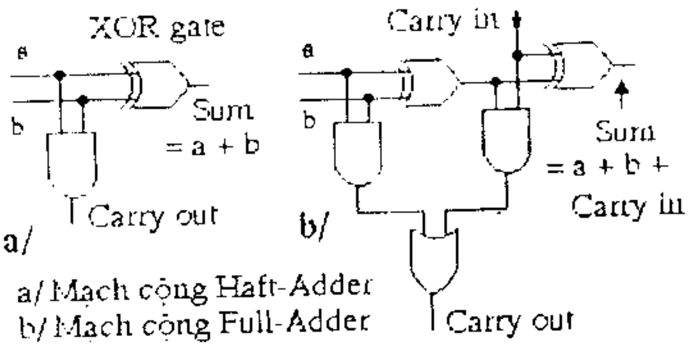
Cho C = 0 và phân tích tương tự như trên ta sẽ thấy rằng các bit đưa vào sẽ được dịch trái một vị trí và đưa ra lối ra.

(Chú ý : Về hướng dịch sang trái hay sang phải là theo hình 3-14, trong hình này trật tự các bit tính từ trái qua phải, ngược lại với trật tự mà chúng ta thường ghi trên giấy như $D_7D_6... D_0$, $S_7S_6... S_0$).

g. Bộ cộng (adder)

Các mạch thực hiện phép cộng là những phần rất căn bản của mọi CPU. Hình 3-14a là mạch tính tổng 2 bit, cho ra tổng (sum) và số nhớ (carry-out) để có thể dùng cho phép cộng 2 bit ở hàng cao hơn, nó còn được gọi là Haft-Adder. Bộ cộng này không có lối vào cho số nhớ Carry-in.

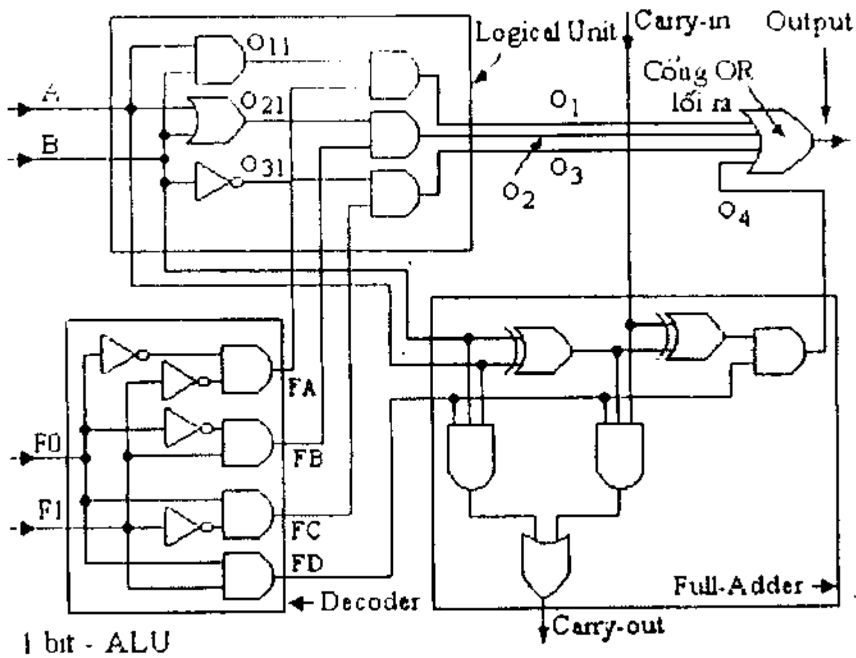
Hình 3-14b là mạch tính tổng 2 bit vào và bit nhớ carry-in, cho ra tổng-sum và số nhớ - carry-out để có thể dùng cho phép cộng bit bậc cao hơn, nó còn được gọi là Full-Adder. Bộ cộng này có lối vào carry-in, nó được xây dựng từ 2 bộ Haft - Adder. Để xây dựng các bộ cộng với từ dài hơn, chẳng hạn từ 16 bit, chỉ cần lặp lại mạch điện của hình 3-14b 16 lần. Bit nhớ carry-out của một hàng bit được sử dụng làm bit carry-out cho việc cộng 2 bit của hàng cao hơn nó một bậc. Đầu vào carry-out của bit bậc thấp nhất được nối với 0. Một bộ cộng như vậy được gọi là ripple carry adder (ripple - làm gợn sóng), có tên gọi như vậy bởi vì trong trường hợp cộng 2 số : 1 và 111 ... 111 phép cộng chỉ được hoàn thành khi số nhớ đã đi suốt từ bit thấp nhất đến bit cao nhất và điều này làm chậm phép tính. Người ta cũng xây dựng các bộ cộng không có nhược điểm này nhưng chúng phức tạp hơn.



Hình 3-14

h. Bộ số học và logic - ALU (arithmetic logical unit)

Các bộ vi xử lý đều có mạch riêng để thực hiện các phép tính And, Or, Not (thực ra chỉ cần And, Not vì đó cũng là hệ hàm đủ, tuy nhiên khi đó tốc độ thực hiện một số bài toán sẽ chậm hơn) và tính tổng của 2 từ. ALU chỉ cần có những mạch này, vì với chừng ấy mạch người ta có thể lập trình để thực hiện tất cả các phép tính logic và số học. Hình 3-15 là một ví dụ đơn giản về mạch tính toán



Hình 3-15

số học - ALU. Tùy theo tín hiệu điều khiển chọn chức năng F0 và F1 là 00, 01, 10 hay 11, nó sẽ thực hiện một trong 4 chức năng sau:

- A AND B
- A OR B
- NOT B (đảo B)
- A + B

Góc dưới bên trái của hình vẽ là bộ giải mã 2 bit (decoder), tuỳ thuộc các giá trị điều khiển đầu vào F0 và F1 mà một trong 4 dây ra FA, FB, FC, FD có mức logic 1, còn 3 dây kia có mức logic 0.

Khối ở góc trên bên trái là khối logic có nhiệm vụ tính A AND B, A OR B, và NOT B. Khối Full -Adder ở góc dưới bên phải hình vẽ thực hiện tính A+B, kể cả việc xử lý số nhớ đưa vào carry-in. Bốn phép tính này cùng được thực hiện nhưng chỉ có một trong bốn kết quả được đưa ra đầu ra Output, bởi vì tín hiệu điều khiển sẽ làm cho 3 trong 4 đầu vào của cổng OR lối ra có giá trị 0, do đó giá trị ra Output sẽ bằng giá trị tại đầu vào còn lại, là kết quả của một trong bốn phép tính (tất nhiên kết quả này có thể là 1 hoặc 0).

Bằng cách sử dụng n mạch ALU như trong hình 3 - 15, đưa carry-out của ALU cộng hai bit bậc i vào đầu vào carry-in của ALU cộng hai bit bậc i+1, chúng ta có thể tạo ra một ALU rộng n bit thực hiện được các phép tính AND, OR, NOT và cộng trên các toán hạng n bit.

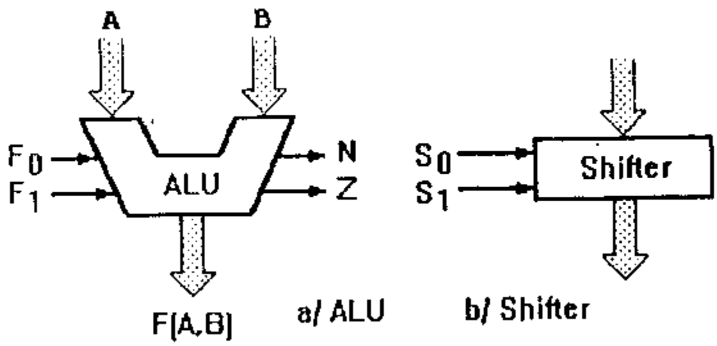
Tuy nhiên trong thực tế, thao tác truyền qua một toán hạng từ lối vào ALU ra lối ra là rất cần thiết và chúng ta sẽ thấy ứng dụng của nó sau này. Để cho cấu trúc của ALU gọn nhẹ, đơn giản, phép toán A OR B được thay thế bởi phép truyền qua A. Chúng ta sẽ nghiên cứu một ALU có thể tính A+B, A AND B, A cũng như $\overline{A}$. ALU cũng có thể có các đầu ra trạng thái, các đầu ra trạng thái điển hình là các đầu ra bằng 1 khi kết quả phép tính là âm, bằng 0 hoặc có carry-out ở bit bậc cao nhất hoặc khi có sự tràn số. Sơ đồ khối ALU trên hình 3-16a có hai đầu ra trạng thái là: N - chỉ báo rằng kết quả ra của ALU là âm (negative) và Z - chỉ báo rằng kết quả ra của ALU bằng không (zero). Thực chất, bit N chỉ là copy của bit có bậc cao nh t của kết quả đưa ra. Bit Z là NOR của tất cả các bit của kết quả đưa ra. Mặc dầu một số ALU có thể thực hiện thao tác dịch (shift) nhưng từ trước tới nay người ta vẫn thường xây dựng đơn vị thực hiện phép dịch riêng biệt. Mạch điện này có thể dịch một số nhiều bit đưa vào sang trái hoặc phải 1 bit hay là không dịch bit nào cả. Vì bộ ghi dịch luôn bổ trợ một cách có hiệu quả cho ALU trong rất nhiều bài toán như nhân đôi (dịch trái), chia đôi (dịch phải), quay vòng, dịch chuyển số liệu... nên sơ đồ này thường được đề

cập khi bàn về ALU. Hình 3-16b là ký hiệu của bộ ghi dịch.

Các chip như trên hình vẽ 3-15 hay 3-16a đã từng có mặt trên thị trường với tên gọi bit-slices (slice - miếng mỏng, lát). Sử dụng các chip này song song với nhau người thiết kế có thể xây dựng nên ALU có độ rộng tuỳ ý, thực hiện các tính toán trên byte, word. Trên thị trường cũng có các chip ALU chứa nhiều bộ ALU kiểu bit-slices, do đó việc thiết kế được đơn giản đi nhiều, số lượng chip cũng giảm. Các chip này cũng có thể kết hợp lại để tạo thành các ALU nhiều bit hơn.

i. Bộ tạo tín hiệu thời gian (clock)

Trong rất nhiều mạch điện số, trật tự xảy ra các sự kiện là hết sức quan trọng. Đôi khi một sự kiện nh t thiết phải xảy ra trước sự kiện khác, có khi 2 sự kiện lại phải xảy ra đồng thời. Để thực hiện được mối quan hệ về mặt thời gian như mong muốn, người thiết kế phải sử dụng nhiều mạch số sử dụng đồng hồ để cung cấp tín hiệu đồng bộ các quá trình.



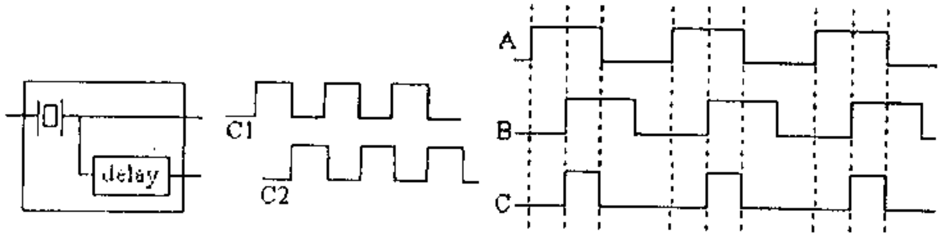
Hình 3-16

Trong lĩnh vực kỹ thuật số, đồng hồ là một mạch điện phát ra chuỗi xung điện, trong đó độ rộng của mỗi xung cũng như chu kỳ lặp lại xung rất ổn định và chính xác. Chu kỳ xung cũng còn được gọi là thời gian chu kỳ đồng hồ. Tần số xung nằm trong khoảng từ 1 đến 100MHz, tương ứng có chu kỳ đồng hồ là 1μs cho đến 10ns. Để đạt được độ chính xác cao và ổn định, người ta thường sử dụng máy phát thạch anh (crystal oscilator).

Trong máy tính, nhiều sự kiện có thể xảy ra trong một chu kỳ máy, ví dụ như các bước để thực hiện một vi lệnh. Nếu các sự kiện này phải xảy ra theo một trật tự nhất định thì chu kỳ máy phải được chia ra thành các chu kỳ con (subcycles). Trong ví dụ trên, mỗi chu kỳ con để thực hiện một bước của vi lệnh

trong một trật tự đã được xác định, ví dụ như kết quả tính toán trong ALU phải có và ổn định trước khi đưa ra bộ nhớ.

Người ta thường sử dụng một giải pháp rất hay là bổ sung thêm một mạch điện, mạch này nối với đường tín hiệu đồng hồ chính qua một bộ phận làm trễ (delay), như vậy có ngay một tín hiệu đồng hồ thứ hai dịch pha so với tín hiệu đồng hồ chính. (Xem hình 3-17a).



a) Đồng hồ

b) Giản đồ thời gian của tín hiệu đồng hồ

c) Sự sinh tín hiệu đồng không cân xứng

Hình 3-17

Giản đồ thời gian trên hình 3-17b cho thấy có 4 tín hiệu thời gian cho các sự kiện khác nhau:

1. Sườn dương của C1.
2. Sườn âm của C1.
3. Sườn dương của C2.
4. Sườn âm của C2.

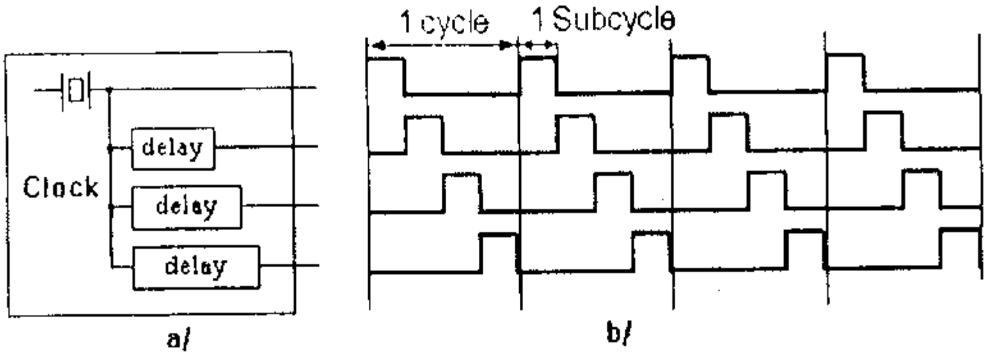
Bằng cách gắn các sự kiện với các sườn xung khác nhau, người ta thực hiện được việc phân bổ thứ tự của các sự kiện. Nếu hệ thống cần trên 4 tín hiệu đồng hồ trong một chu kỳ máy, thì các đường tín hiệu đồng hồ phụ sẽ được bổ sung vào, mỗi đường có độ trễ khác nhau.

Trong một số mạch điện người ta cần các mức tín hiệu đồng hồ kéo dài trong một khoảng thời gian chứ không phải trong các thời điểm rời rạc. Thí dụ một số sự kiện có thể được phép xảy ra trong khoảng thời gian C1 ở mức cao, chứ không phải chính xác ở sườn dương của C1. Nếu cần tới trên 2 tín hiệu như vậy người ta có thể dùng nhiều đường tín hiệu đồng hồ hoặc cho các trạng thái có mức cao của 2 tín hiệu đồng hồ trùm phủ một phần lên nhau (theo trục thời gian). Trong trường hợp dùng 2 tín hiệu đồng hồ trùm phủ một phần lên nhau

ta có các khoảng thời gian tách biệt sau :

$\overline{C1}$ AND $\overline{C2}$, C1 AND C2, C1 AND $\overline{C2}$, C1 AND C2

Trên hình 3-17c, tín hiệu c) có được bằng cách thực hiện AND các tín hiệu a) và b). Cuối cùng trên hình 3 - 18 đưa ra một đồng hồ có 4 lối ra và giản đồ thời gian rất cần để thực hiện một vi lệnh sẽ bàn đến trong phần mức vi chương trình.



Hình 3-18

Lối ra trên cùng là lối ra cơ sở, các lối ra còn lại nhận được từ lối ra này bằng cách chèn các bộ trễ (delay) có độ trễ khác nhau vào lối ra. Tín hiệu đồng hồ cơ sở, đường trên cùng của hình 3-18b, có độ rộng xung bằng 1/4 chu kỳ. Các tín hiệu ra khác đã được làm trễ một, hai và 3 lần độ rộng xung, kết quả là mạch điện chia mỗi chu kỳ thành 4 chu kỳ con dài bằng nhau.

IV. MỨC VI CHƯƠNG TRÌNH

Người ta không xác định thật rõ ràng ranh giới giữa phần cứng và phần mềm, hơn nữa nó cũng không ngừng dịch chuyển. Trong những máy tính đầu tiên tất cả các chỉ thị thực hiện các phép toán số học, logic, dịch, so sánh, lặp... đều được thực hiện bằng phần cứng. Với mỗi chỉ thị có một mạch điện phần cứng cụ thể thực hiện chỉ thị đó. Về nguyên tắc, người ta có thể tháo ốc, mở nắp máy và chỉ ra những mạch điện tử thực hiện một chỉ thị cụ thể, chẳng hạn chỉ thị chia.

Trong máy tính nhiều mức hiện đại, người ta không thể chỉ ra mạch thực hiện phép chia bởi vì không có mạch thực hiện phép chia. Tất cả các chỉ thị có thể sử dụng được ở mức lệnh máy nhị phân (thí dụ các chỉ thị thực hiện các phép

toán số học, logic, dịch, so sánh, lặp) được thực hiện từng bước một bằng một trình thông dịch chạy ở mức vi chương trình. Việc tìm ra mạch thực hiện phép chia ngày nay tương đương với việc lấy ra danh sách các vi chương trình và tìm phần vi chương trình thực hiện chỉ thị chia. Người ta biết rằng, chỉ bằng bộ cộng (có cả chức năng trừ) có trong ALU người ta có thể lập trình để thực hiện các phép tính số học. Tương tự, trong ALU chỉ có các sơ đồ logic của hệ hàm đủ (And, Or, Not) hoặc (And, Not) và người ta có thể lập trình để thực hiện các phép tính logic.

Mặc dầu mọi chương trình ở mọi mức đều có thể thi hành bằng trình thông dịch và mặc dầu chính trình thông dịch này lại có thể được thi hành bởi một trình thông dịch khác, nhưng trình tự như vậy không thể nào cứ kéo dài mãi. Ở mức dưới cùng nhất thiết phải có một cái máy phần cứng vật lý, chế tạo bởi các vi mạch (IC), nguồn nuôi và các thành phần "cứng" tương tự như thế. Những thành phần "cứng" như vậy là đối tượng mà chúng ta đã nghiên cứu ở phần trước. Phần này nghiên cứu việc các thành phần phần cứng này được điều khiển bởi vi chương trình như thế nào và vi chương trình thông dịch mức lệnh máy nhị phân như thế nào.

Bởi vì kiến trúc của mức vi chương trình, còn được gọi là vi kiến trúc được xác định bởi phần cứng cho nên nó thường rất sơ đẳng và rất khó lập trình, chẳng hạn việc định thời (timing) cho các thao tác phần cứng.

Mức vi chương trình có một chức năng chuyên biệt là thi hành các trình thông dịch cho các máy ảo khác. Mục đích thiết kế này tự nhiên dẫn tới một sự tổ chức được tối ưu hóa ở mức cao, hướng tới việc lấy chỉ thị, giải mã chỉ thị và thi hành các chỉ thị của mức lệnh máy nhị phân. Trong một số trường hợp thực hiện các công việc trên ngay cả đối với các chỉ thị phức tạp hơn.

Phần này sẽ khảo sát các vấn đề liên quan tới việc tổ chức và thiết kế mức vi chương trình cũng như những sự cân nhắc lựa chọn các phương pháp.

Chúng ta sẽ bắt đầu nghiên cứu mức vi chương trình bằng cách ôn lại tóm tắt các thành phần cơ bản xây dựng nên máy tính đã được nghiên cứu ở phần trên, chúng là một phần của kiến trúc của mức vi chương trình, là mối quan tâm của những người lập vi chương trình. Sau đó sẽ tiến đến nghiên cứu phần chính yếu là các chỉ thị phức tạp hơn có thể được xây dựng nên từ một dãy các chỉ thị sơ cấp như thế nào. Chủ đề này sẽ được thảo luận chi tiết và minh họa bằng ví dụ cụ thể. Sau đó chúng ta sẽ khảo sát các yếu tố khác nhau cần phải tính đến

khi thiết kế mức vi chương trình của một máy tính, nhằm hiểu tốt hơn tại sao người ta đã làm như vậy. Chúng ta cũng xem các cách nâng cao hiệu suất của máy tính.

1. Các vấn đề chính của mức logic số có liên quan

Công việc của người lập vi chương trình là viết chương trình điều khiển các thanh ghi, các bus, các ALU, bộ nhớ và các phần tử phần cứng khác của máy. Các thiết bị này đã được nghiên cứu khá chi tiết ở phần trước, chúng ta chỉ ôn lại một cách tóm tắt và đưa thêm một số chi tiết liên quan đến phần này, để có thể hiểu được thực chất công việc của người lập vi chương trình.

1.1. Thanh ghi

Thanh ghi là một thiết bị có khả năng chứa thông tin. Có thể hiểu đó là một ô nhớ nhiều bit đứng độc lập. Mức vi chương trình luôn luôn có một số thanh ghi để giữ các thông tin cần thiết cho việc xử lý chỉ thị hiện đang được thông dịch. Về mặt khái niệm các thanh ghi cũng giống như bộ nhớ chính, sự khác nhau là về mặt vật lý. Các thanh ghi được đặt ngay bên trong bộ xử lý, vì vậy việc đọc/ghi thông tin ở đó nhanh hơn so với đọc/ghi các từ trong bộ nhớ, các từ của bộ nhớ luôn nằm ngoài chip xử lý. Những máy tính lớn và đắt tiền thường có nhiều thanh ghi hơn các máy nhỏ và rẻ, các máy nhỏ và rẻ buộc phải sử dụng bộ nhớ chính để chứa các kết quả trung gian. Trong một số máy tính, một nhóm các thanh ghi được đánh số 0, 1, 2... n-1 có thể sử dụng được ở mức vi chương trình và chúng được gọi là bộ nhớ tạm hay bộ nhớ nháp (scratchpad storage).

Thanh ghi được đặc trưng bằng số bit mà nó có thể chứa. Hình 3-19 là một thanh ghi 16 bit, với các bit được đánh số thứ tự (bit number) từ phải qua trái.

Bit number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	1	1	0	0	1	0	1	1	0	0	1	0

Hình 3-19

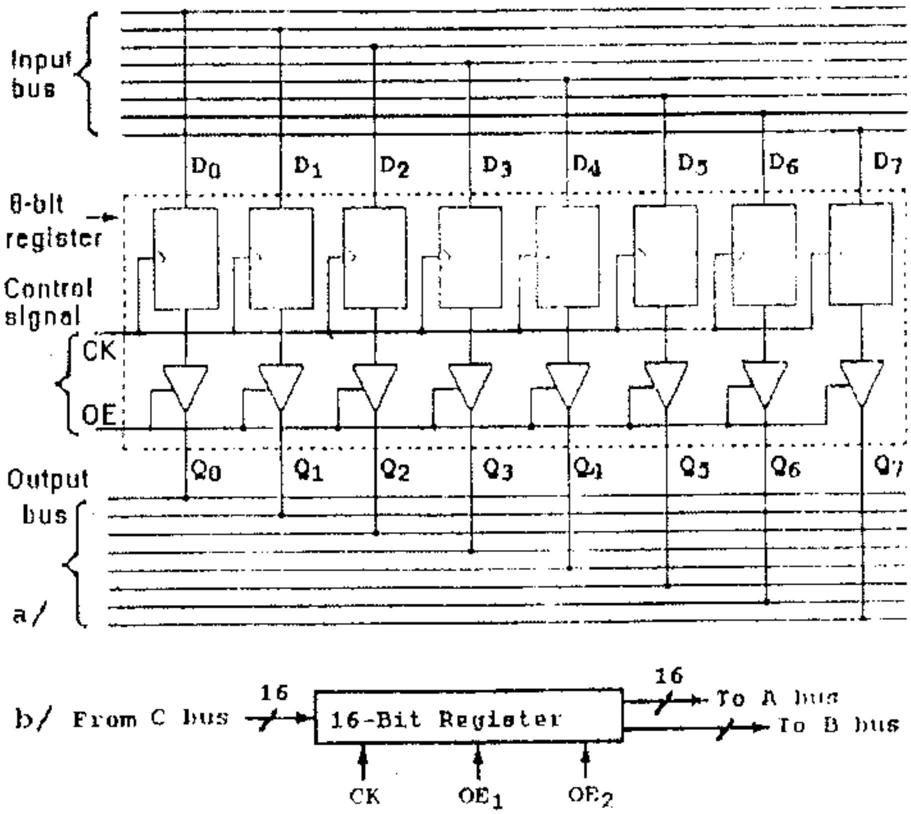
Quá trình đọc thông tin từ một thanh ghi không làm ảnh hưởng tới nội dung của thanh ghi. Nói một cách khác, khi một thanh ghi bị đọc, có một bản copy nội dung của nó được tạo ra, còn giá trị gốc không bị suy chuyển vẫn để trong thanh ghi.

1.2. Bus

Bus là tập hợp các đường dây được sử dụng để truyền song song các tín hiệu (bit). Ví dụ dùng để copy nội dung thanh ghi này sang một thanh ghi khác. Người ta sử dụng bus vì việc truyền cùng một lúc nhiều bit đi song song nhanh hơn truyền lần lượt từng bit.

Bus có thể là một hoặc hai chiều. Bus một chiều chỉ có thể truyền dữ liệu theo một chiều, trong khi đó bus hai chiều có thể truyền dữ liệu đi theo cả hai chiều, nhưng không thể truyền theo cả hai chiều cùng một lúc được. Bus một chiều thường được sử dụng để nối hai thanh ghi, trong đó một thanh ghi luôn luôn là nguồn, còn thanh ghi kia luôn luôn là đích. Bus hai chiều hay được sử dụng khi một thanh ghi bất kỳ trong tập hợp các thanh ghi có thể là nguồn và một thanh ghi bất kỳ khác là đích.

Nhiều thiết bị có khả năng nối và tách chính nó về mặt điện ra khỏi bus mà nó được nối kết về mặt vật lý, việc nối và cắt có thể được thực hiện trong vài nanô giây. Bus mà các thiết bị của nó có tính chất này được gọi là bus 3 trạng thái, bởi vì mỗi đường dây có thể có giá trị 0 hay 1 hoặc là bị tách ra khỏi hệ thống(trạng thái trở kháng cao hay trạng thái treo). Các bus 3 trạng thái thường được sử dụng khi cần nối nhiều thiết bị vào bus, mỗi thiết bị này đều có thể đổ thông tin lên bus. Trong hầu hết các vi kiến trúc, một số thanh ghi được nối với một hoặc một vài bus vào và một hoặc một vài bus ra. Hình 3-18a mô tả một thanh ghi 8 bit được nốivới một bus vào và một bus ra. Thanh ghi này gồm có 8 flip-flop loại D, mỗi flip-flop nối với bus ra qua một bộ đệm, mỗi flip-flop chứa 1 bit thông tin. Thanh ghi có hai tín hiệu điều khiển, CK (clock) điều khiển nạp vào thanh ghi và OE (output Enable) - cho phép thanh ghi đưa dữ liệu ra, cả hai tín hiệu này nối với mọi flip-flop. Lúc bình thường cả hai tín hiệu này nằm ở trạng thái không tích cực, khi đó thanh ghi bị tách ra khỏi hệ thống. Flip-flop loại D có lối vào dữ liệu là D và lối ra là Q và Q = D khi CK ở mức tích cực, đây là loại ô nhớ SRAM. Xét riêng từng tín hiệu, khi tín hiệu CK không tích cực, nội dung của thanh ghi không bị ảnh hưởng của các tín hiệu trên bus vào (input bus). Khi CK được đặt tích cực, thanh ghi được nạp từ bus vào. Khi tín hiệu OE bị đặt không tích cực, thanh ghi bị tách khỏi bus ra (output bus) và có thể coi là không còn ảnh hưởng gì đối với các thanh ghi khác trên bus. Khi OE được đặt tích cực, nội dung của thanh ghi được đưa lên bus ra.



Hình 3-20

Nếu một thanh ghi R khác có bus vào nối với bus ra của thanh ghi này, thì có thể truyền thông tin từ thanh ghi này tới thanh ghi R. Để làm việc này, phải đặt OE mức tích cực và giữ mức này đủ lâu sao cho tín hiệu mà thanh ghi đưa ra bus ra trở nên ổn định. Sau đó đường dây tín hiệu CK của thanh ghi R phải được đặt tích cực để nạp thông tin trên bus cho R. Thao tác mở cổng một thanh ghi vào một bus để cho một thanh ghi khác có thể nạp giá trị từ bus vào thường xảy ra ở mức vi chương trình, chúng ta sẽ nghiên cứu vấn đề này ở một phần dưới đây. Hình 3-18b là một ví dụ thứ hai về thanh ghi và bus, hình vẽ mô tả một thanh ghi 16 bit có hai bus ra, mỗi bus ra được điều khiển bằng một tín hiệu OE khác nhau.

1.3. Bộ nhớ đệm

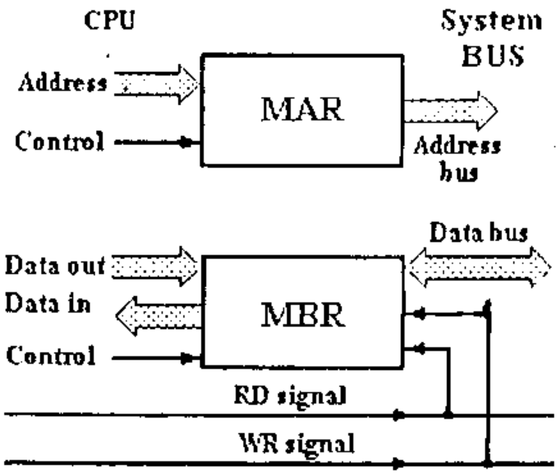
Các bộ xử lý cần phải có khả năng đọc dữ liệu từ bộ nhớ và ghi dữ liệu vào bộ nhớ. Hầu hết các máy tính có một bus địa chỉ (address), một bus dữ liệu

(data) và một bus điều khiển (control) để truyền thông giữa CPU và bộ nhớ.

Để đọc từ bộ nhớ, CPU đặt một địa chỉ lên bus địa chỉ và thiết lập các tín hiệu điều khiển một cách phù hợp, chẳng hạn đặt tín hiệu RD (read - đọc) tích cực, sau đó bộ nhớ đưa dữ liệu mà nó bị yêu cầu lên bus dữ liệu. Trong một số máy tính, việc đọc ghi bộ nhớ là đồng bộ, nghĩa là bộ nhớ phải đáp ứng được yêu cầu đọc/ghi đối với nó trong một khoảng thời gian cố định. Trong một số máy tính khác, bộ nhớ có thể chậm bao nhiêu cũng được, để thực hiện được việc này phải bổ sung thêm cho bus một đường tín hiệu điều khiển, khi nào bộ nhớ hoàn thành công việc nó báo cho đơn vị đã có yêu cầu dữ liệu biết.

Việc ghi vào bộ nhớ được thực hiện tương tự. CPU đặt dữ liệu cần ghi vào bộ nhớ lên bus dữ liệu và đặt địa chỉ ô nhớ sẽ chứa dữ liệu lên bus địa chỉ, sau đó đặt dây tín hiệu WR (write) lên mức tích cực.

Một thao tác truy cập bộ nhớ (đọc/ghi) nhìn chung thường lâu hơn nhiều thời gian cần thiết để thực hiện một chỉ thị. Hệ quả là vi chương trình phải duy trì các giá trị trên các bus địa chỉ và bus dữ liệu trong suốt thời gian thi hành một số vi lệnh. Để cho công việc này được đơn giản và thuận tiện, người ta thường dùng hai thanh ghi là MAR (memory address register: thanh ghi địa chỉ bộ nhớ) và thanh ghi MBR (memory buffer register: thanh ghi đệm bộ nhớ), chúng sẽ điều khiển bus địa chỉ và bus dữ liệu. Để tiện cho việc nghiên cứu, chúng ta sắp xếp các bus như trên hình 3-21. Cả hai thanh ghi nêu trên nằm giữa CPU và bus hệ thống (system bus). Bus địa chỉ là một chiều ở cả hai bên thanh ghi MAR, nó được CPU nạp giá trị lên khi tín hiệu control được đặt mức tích cực.



Hình 3-21

Đầu ra của MAR nối với các đường dây của bus địa chỉ hệ thống luôn luôn được đặt tích cực (có trường hợp chỉ được đặt tích cực khi ghi hoặc đọc). Đường dây control của MBR làm cho dữ liệu từ MBR thông qua bus "Data in" nạp vào trong CPU. Đường dây "Data out" thực hiện theo chiều ngược lại. Bus dữ liệu của hệ thống là hai chiều, đưa dữ liệu từ MBR ra khi tín hiệu WR tích cực và nạp vào MBR khi tín hiệu RD là tích cực.

2. Một ví dụ về vi kiến trúc

Tới lúc này chúng ta đã nghiên cứu tất cả các thành phần cơ bản để xây dựng nên mức vi chương trình, đã đến lúc chúng ta xem xét việc chúng được nối với nhau như thế nào. Trong lĩnh vực này có rất ít nguyên lý chung, vì vậy chúng ta sẽ nghiên cứu vấn đề thông qua một ví dụ chi tiết.

2.1. Đường dữ liệu (Data path)

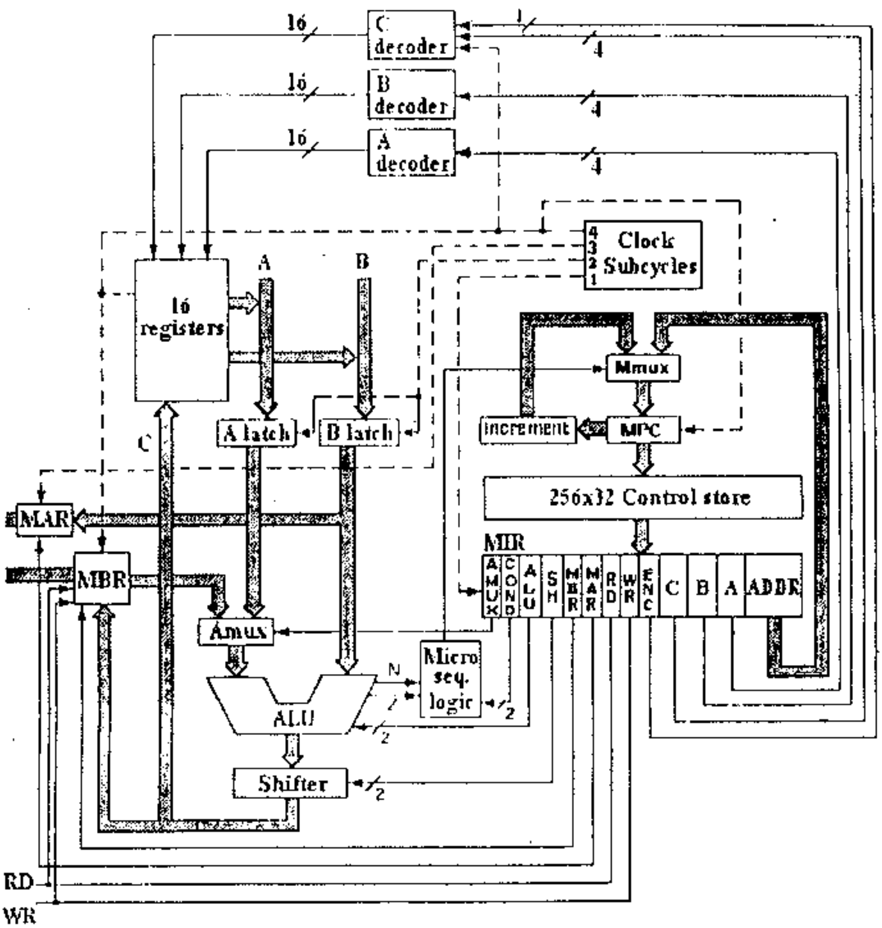
Đường dữ liệu là một phần của CPU, nó có chứa ALU, các đầu vào và đầu ra. Trên hình 3-22 là một đường dữ liệu của một vi kiến trúc mà chúng ta lấy làm thí dụ: nó bao gồm 16 thanh ghi (đánh số từ 1 - 16) giống nhau, được chúng ta gắn nhãn PC, AC, SP... và có thể gọi theo tên nhãn này hoặc theo số thứ tự, chúng tạo nên một bộ nhớ tạm thời chỉ có thể truy cập được ở mức vi chương trình. Mỗi thanh ghi có thể đưa nội dung của mình ra ngoài, lên một hay cả hai bus bên trong, đó là bus A và bus B. Mỗi thanh ghi có thể được nạp từ một bus thứ ba là bus C. Các bus A và B nạp dữ liệu vào ALU rộng 16 bit, ALU đó có thể thực hiện 4 chức năng : A + B, A AND B, A và NOT A. Hai đường điều khiển F_0 và F_1 định rõ chức năng sẽ được ALU thực hiện. ALU sinh ra hai bit trạng thái dựa trên kết quả ra hiện thời của nó: N có giá trị 1 khi kết quả ra là âm, Z có giá trị 1 khi kết quả ra bằng 0 (zêro). Đầu ra của ALU đưa vào một thanh ghi dịch - shifter, nó có thể dịch các bit một vị trí theo cả hai hướng hoặc không dịch. Có thể thực hiện dịch trái hai bit một thanh ghi R, bằng cách tính R+R trong ALU, sau đó dịch tổng một bit nữa sang trái bằng thanh ghi dịch.

Cả bus A lẫn bus B đều không nạp trực tiếp vào ALU, thay vào đó mỗi bus nạp vào một thanh ghi chốt, đến lượt mình các thanh ghi chốt nạp vào ALU. Cần phải sử dụng các thanh ghi chốt bởi vì ALU là một mạch tổ hợp, nó liên tục tính và đưa ra kết quả theo các giá trị vào hiện thời và theo mã chức năng.

Sự tổ chức này có thể gây ra những vấn đề, thí dụ khi đang tính A:=A+B. Vì thanh ghi A đang được chứa kết quả vào nên giá trị trên bus A bắt đầu thay đổi, nó sẽ làm cho đầu ra của ALU và do đó làm cho bus C cũng thay đổi theo. Hệ quả là giá trị sai có thể sẽ được chứa vào trong thanh ghi A. Nói cách khác,

trong phép gán A: =A+B, A ở vế phải là giá trị ban đầu của A, không phải là sự trộn lẫn từng bit một của các giá trị mới và cũ. Bằng cách chèn các thanh ghi chốt A latch và B latch vào các bus A và B, chúng ta có thể cho các thanh ghi chốt này chứa và đưa đến hai đầu vào ALU các giá trị ban đầu ở bus A và bus B. Vì thế ALU được bảo vệ không bị thay đổi do các thay đổi trên các bus A và B khi một giá trị mới đang được chứa vào bộ nhớ tạm (là tập 16 thanh ghi ở phía trên của hình). Các tín hiệu L_0 và L_1 điều khiển việc nạp giá trị trên bus A và bus B vào các thanh ghi chốt.

Để truyền thông với bộ nhớ chúng ta đã bổ sung thêm hai thanh ghi MAR và MBR trong vi kiến trúc. MAR có thể được nạp từ thanh ghi chốt B song song với một thao tác của ALU. Đường dây Mo điều khiển việc nạp của MAR.



Hình 3-22

Trong các thao tác ghi vào bộ nhớ chính, MBR được nạp giá trị từ đầu ra của thanh ghi dịch, giá trị này cũng có thể đồng thời được chứa vào một trong các thanh ghi của bộ nhớ tạm, M1 điều khiển việc nạp của MBR từ đầu ra của thanh ghi dịch, RD và WR điều khiển việc đọc và ghi bộ nhớ. Khi đọc, dữ liệu đọc được từ bộ nhớ qua MBR có thể có mặt ở đầu vào bên trái của ALU thông qua một bộ dồn kênh, trên sơ đồ ghi là Amux. Đường dây điều khiển Ao xác định sẽ nạp vào ALU từ thanh ghi chốt A latch hay từ MBR.

Vi kiến trúc trên hình vẽ 3-22 tương tự vi kiến trúc của một số chip đã từng được bán trên thị trường.

2.2. Vi chỉ thị (Microinstruction)

Để điều khiển đường dữ liệu trên hình 3-22 chúng ta cần 59 tín hiệu, chúng có thể được chia thành 9 nhóm chức năng như mô tả dưới đây:

- 16 tín hiệu để điều khiển việc nạp cho bus A từ bộ nhớ tạm (có 16 thanh ghi)
- 16 tín hiệu để điều khiển việc nạp cho bus B từ bộ nhớ tạm
- 16 tín hiệu để điều khiển việc nạp cho bộ nhớ tạm từ bus C
- 2 tín hiệu để điều khiển các thanh ghi chốt A và B
- 2 tín hiệu để điều khiển chức năng ALU
- 2 tín hiệu để điều khiển bộ dịch
- 2 tín hiệu để điều khiển MAR và MBR (M0, M1)
- 2 tín hiệu để chỉ rõ việc đọc bộ nhớ và ghi vào bộ nhớ.
- 1 tín hiệu để điều khiển Amux.

Khi đã có các giá trị của 59 tín hiệu, chúng ta có thể thực hiện một chu kỳ của đường dữ liệu. Một chu kỳ bao gồm việc mở cổng cho các giá trị trong bộ nhớ tạm đi vào bus A và bus B, chốt chúng lại trong hai thanh ghi chốt bus A latch và B latch, cho các giá trị từ các thanh ghi chốt chạy qua ALU và shifter, cuối cùng là việc chứa các kết quả vào trong bộ nhớ tạm hoặc vào MBR. Ngoài ra MAR cũng có thể được nạp, sau đó một chu kỳ bộ nhớ sẽ bắt đầu.

Chúng ta sẽ lấy một ví dụ gần đúng: sử dụng một thanh ghi điều khiển 59 bit, mỗi bit dùng cho một tín hiệu điều khiển. Bit bằng 1 có nghĩa là tín hiệu được đặt tích cực, còn bằng 0 nghĩa là không tích cực. Tuy nhiên nếu chúng ta chấp nhận bổ sung thêm một số mạch điện, thì có thể giảm đi được nhiều bit mà chúng ta cần dùng để điều khiển đường dữ liệu. Chúng ta bắt đầu xem xét từ 16 bit để điều khiển việc đưa dữ liệu vào bus A, với 16 bit này có thể chọn một trong 216 bit tổ hợp các thanh ghi nguồn, nhưng chỉ có 16 trong số các tổ

hợp này là được phép, vì chúng ta cũng chỉ có 16 thanh ghi. Chính vì vậy chúng ta có thể lập mã (encode) thông tin điều khiển bus A chỉ bằng 4 bit và dùng một bộ giải mã để sinh ra 16 tín hiệu điều khiển. Đối với bus B, bus C cũng như vậy.

Như vậy cho tới nay đã tiết kiệm được 3x12 bit = 36 bit, bây giờ chúng ta chỉ còn cần 23 bit (59-36 = 23bit) điều khiển để vận hành đường dữ liệu. Hai tín hiệu Lo và L1 luôn luôn cần đến ở những thời điểm nhất định, vì vậy chúng sẽ được một đồng hồ cung cấp, chúng ta bớt đi được hai bit điều khiển, còn lại 21 bit. Có một tín hiệu điều khiển nữa không bắt buộc phải có nhưng thường là hữu ích, đó là tín hiệu cho phép hoặc không cho phép chứa giá trị trên bus C vào bộ nhớ tạm. Trong một số tình huống người ta chỉ muốn thực hiện một thao tác ALU để sinh ra các tín hiệu trạng thái N và Z chứ không cần lưu trữ kết quả. Với bit thêm vào này, ta đặt tên là ENC (enable C), chúng ta có thể chỉ ra rằng có cất giá trị trên bus C (cho ENC = 1) hay không cất (cho ENC = 0). Như vậy tới thời điểm này chúng ta có thể điều khiển đường dữ liệu bằng một con số 22 bit.

AMUX : điều khiển lối vào bên trái của ALU : 0 = A latch, 1 = MBR
ALU : chức năng ALU : 0 = A+B, 1= A AND B, 2 = A, 3 =/ A
SH : chức năng thanh ghi dịch : 0 = no sshift, 1 = right, 2= left
MBR : nạp cho MBR từ thanh ghi dịch : 0 = không nạp, 1 = nạp
MAR : nạp cho MAR từ thanh ghi chốt B : 0 = không nạp, 1 = nạp
RD : yêu cầu đọc bộ nhớ : 0 = không đọc, 1= nạp cho MBR từ bộ nhớ
WR : yêu cầu ghi bộ nhớ : 0 = không ghi, 1 = ghi MBR vào bộ nhớ
ENC : điều khiển việc chứa vào bộ nhớ tạm : 0 = không chứa, 1 = chứa
C : chọn thanh ghi để chứa vào (nếu ENC=1) : 0=PC, 1=AC,..
B : chọn nguồn cho bus B : 0 = PC, 1=AC...
A : chọn nguồn cho bus A : 0=PC, 1=AC...

Bước kế tiếp trong việc thiết kế vi kiến trúc là tạo ra một khuôn dạng vi chỉ thị chứa 22 bit. Một khuôn dạng vi chỉ thị bao gồm 13 trường, trong đó 11 trường mô tả ở trên và hai trường bổ sung thêm là COND và ADDR, sẽ mô tả ở mục d. Trật tự của các trường là hoàn toàn tuỳ ý, vì thứ tự thực hiện lại do các xung đồng hồ có nhiệm vụ định thời quyết định. Điều này dễ hiểu vì các bit của trường điều khiển mới chỉ mở các chốt để xác định đường, hướng truyền dữ liệu cũng như mới xác định trước các nhiệm vụ cần thực hiện

2.3. Việc định thời vi chỉ thị

Mặc dù chúng ta đã thảo luận gần như đầy đủ về việc vi chỉ thị có thể điều

khiển đường dữ liệu như thế nào trong một chu kỳ, cho tới điểm này chúng ta vẫn còn bỏ qua một vấn đề là định thời, đó là việc định trình tự xảy ra các sự kiện trong quá trình thực hiện một vi chỉ thị.

Một chu kỳ ALU cơ sở bao gồm việc thiết lập giá trị cho các thanh ghi chốt A và B, dành thời gian cho ALU và bộ ghi dịch (shifter) thực hiện công việc của nó và cất các kết quả. Trình tự như vậy là rất hiển nhiên; Nếu chúng ta cứ chứa giá trị trên bus C vào bộ nhớ trước khi các thanh ghi chốt A và B được nạp xong, thì chỉ có rác được chứa chứ không phải là các dữ liệu hữu ích.

Để đạt được việc định đúng trình tự các sự kiện, bây giờ chúng ta đưa ra một đồng hồ 4 pha, tức là một đồng hồ có 4 chu kỳ con, như trên hình vẽ 3-20. Các sự kiện chủ yếu xảy ra trong mỗi một trong số 4 chu kỳ con như sau:

- Nạp vi chỉ thị tiếp theo sẽ được thi hành vào một thanh ghi có tên là MIR-microinstruction register.
- Mở cổng các thanh ghi tạm đi vào bus A và bus B và giữ chúng trong các thanh ghi chốt A và B (A latch và B latch).
- Khi các giá trị đưa vào ALU ổn định, dành thời gian cho ALU và shifter để chúng sinh ra giá trị ra ổn định; nạp Mar nếu cần thiết.
- Khi giá trị ra của bộ ghi dịch (shifter) ổn định, chứa giá trị trên bus C vào bộ nhớ và nạp cho MBR nếu việc này cũng được yêu cầu.

Hình 3-22 là sơ đồ khối chi tiết của vi kiến trúc đầy đủ của chiếc máy mà chúng ta lấy làm thí dụ. Thoạt nhìn nó có vẻ thật đồ sộ nhưng rất đáng để chúng ta nghiên cứu tỉ mỉ. Khi hiểu đầy đủ về từng khối và từng đường dây trên sơ đồ chúng ta sẽ bước vững chắc trên con đường nghiên cứu mức vi chương trình của mình. Sơ đồ khối có hai phần, đường dữ liệu phía bên trái đã được chúng ta bàn luận chi tiết và phần điều khiển phía bên phải sẽ được xem xét tại đây.

Phần tử lớn nhất và quan trọng nhất trong phần điều khiển của máy tính là bộ nhớ điều khiển - control store. Bộ nhớ tốc độ cao và đặc biệt này là nơi giữ các vi chỉ thị. Trong một số máy đó là bộ nhớ chỉ đọc và trong một số máy khác lại là bộ nhớ đọc/ghi. Trong ví dụ của chúng ta các vi chỉ thị có độ rộng 32 bit (22 bit đã được mô tả ở phần trên, ngoài ra còn có thêm trường ADDR rộng 8 bit và trường COND rộng 2 bit) và không gian địa chỉ vi chỉ thị bao gồm 256 từ, vì thế bộ nhớ điều khiển sẽ chiếm tối đa là 256 x 32 = 8192bit.

Bộ nhớ điều khiển và bộ nhớ chính là hoàn toàn khác nhau, bộ nhớ điều khiển chứa vi chương trình, còn bộ nhớ chính chứa chương trình ngôn ngữ máy

thông thường. Giống như mọi bộ nhớ khác, bộ nhớ điều khiển cần một MAR và một MBR. Chúng ta sẽ gọi MAR của bộ nhớ điều khiển là MPC (microprogram counter: con đếm hay con trỏ vi chương trình), bởi vì chức năng duy nhất của nó là trỏ đến vi chỉ thị tiếp theo sẽ được thực hiện còn MBR của bộ nhớ điều khiển là MIR như đã nhắc đến ở trên.

• Chu kỳ con thứ nhất:

Từ hình vẽ 3-22 ta có thể thấy rõ rằng bộ nhớ điều khiển không ngừng chuyển các bit của vi chỉ thị đã được định địa chỉ bởi MPC vào MIR. Tuy nhiên MIR chỉ được nạp trong chu kỳ con thứ nhất, như đã được chỉ ra bằng đường đứt nét nối từ đồng hồ tới nó. Trong khoảng thời gian 3 chu kỳ con khác nó không bị ảnh hưởng, dù có điều gì xảy ra với MPC đi nữa.

• Chu kỳ con thứ hai:

Trong khoảng thời gian của chu kỳ con thứ hai, MIR không thay đổi và các trường khác nhau của MIR bắt đầu điều khiển đường dữ liệu.

Các khối có ghi A decoder và B decoder trên sơ đồ khối làm công việc giải mã 4 ra 16 đường (4 to 16), với 4 đường dây vào mỗi khối được nối với 4 bit của từng trường A hoặc B trong MIR. Một trong 16 đầu ra của mỗi decoder này có mức tích cực, tín hiệu đó sẽ điều khiển các đường dây OE1 và OE2 trong các thanh ghi nối với chúng. Hai trong số 16 thanh ghi tạm được các bộ giải mã A decoder và B decoder chọn sẽ mở cổng ra, đưa nội dung của chúng lên các bus A và bus B.

Đồng hồ kích hoạt các thanh ghi chốt A latch và B latch trong khoảng thời gian chu kỳ con này, chúng sẽ chốt dữ liệu trên bus A và bus B và cung cấp tín hiệu vào ALU ổn định trong thời gian còn lại của chu kỳ. Trong khi dữ liệu đang được mở cổng đi vào các bus A và bus B thì đơn vị Increment trong bộ phận điều khiển tính MPC +1, để chuẩn bị cho việc nạp vi chỉ thị theo trình tự tiếp theo trong chu kỳ tiếp theo. Việc gối đầu hai thao tác này, có thể làm tăng tốc việc thực hiện chỉ thị.

• Chu kỳ con thứ ba:

Trong chu kỳ con thứ ba ALU và bộ ghi dịch (shifter) được dành thời gian để sinh ra kết quả đúng. Trường AMUX của vi chỉ thị xác định giá trị vào nhánh trái của ALU, còn giá trị vào nhánh phải luôn luôn là thanh ghi chốt B. Mặc dầu ALU là một mạch tổ hợp, thời gian mà nó cần để tính tổng được xác định bởi thời gian lan truyền carry, chứ không phải là độ trễ của một cổng thông thường.

Thời gian lan truyền carry tỉ lệ với số bit trong từ. Trong khi ALU và shifter đang tính toán thì MAR được nạp từ bus B nếu trường MAR trong vi chỉ thị bằng 1.

• Chu kỳ con thứ tư:

Trong thời gian chu kỳ con thứ tư và là chu kỳ con cuối cùng, bus C có thể được nạp ngược trở lại vào bộ nhớ tạm và MBR, tuỳ thuộc vào các trường ENC và MBR của vi chỉ thị. Khối có ghi nhãn "C decoder" nhận ở đầu vào các tín hiệu ENC, đường tín hiệu đồng hồ thứ tư và trường C của vi chỉ thị, nó sinh ra 16 tín hiệu điều khiển. Bên trong nó thực hiện giải mã (4 to 16) của trường C của MIR, sau đó thực hiện AND từng tín hiệu này với một tín hiệu nhận được bằng cách AND tín hiệu trên đường dây chu kỳ con thứ tư với ENC. Chính vì thế một thanh ghi của bộ nhớ tạm chỉ được nạp từ bus C nếu 3 điều kiện sau đây được đồng thời thoả mãn :

+ ENC = 1

+ Đang trong chu kỳ con thứ 4

+ Thanh ghi đã được chọn bởi trường C của vi chỉ thị

MBR cũng được nạp trong chu kỳ con thứ tư nếu trường MBR = 1.

Hai tín hiệu điều khiển bộ nhớ RD và WR được đặt tích cực khi các trường tương ứng RD và WR trong MIR là tích cực. Hệ quả là các trường tương ứng của MIR thể hiện như các thanh ghi chốt: giá trị mà nó nhớ luôn được đưa ra, còn đầu vào bị chốt lại, làm cho nội dung mà nó nhớ không thay đổi.

2.4. Sự định trình tự các vi chỉ thị

Chỉ còn lại một vấn đề là vi chỉ thị tiếp theo sẽ được chọn như thế nào. Mặc dầu đôi khi chỉ cần lấy vi chỉ thị kế tiếp theo thứ tự là đủ, nhưng cũng cần một cơ chế cho phép thực hiện việc nhảy có điều kiện trong vi chương trình. Chính vì lý do này mà trong mỗi vi chỉ thị chúng ta đã đưa vào thêm hai trường: ADDR là địa chỉ của vi chỉ thị có thể sẽ được thực hiện tiếp sau vi chỉ thị hiện thời và trường COND, nó xác định sẽ lấy vi chỉ thị tiếp theo từ địa chỉ MPC +1 hay từ địa chỉ ADDR trong bộ nhớ điều khiển.

Mọi vi chỉ thị đều có thể bao gồm cả việc nhảy có điều kiện. Người ta làm như vậy bởi vì nhảy có điều kiện rất thường gặp trong các vi chương trình và điều đó cho phép mọi vi chỉ thị đều có hai vi chỉ thị kế tiếp để lựa chọn thực hiện. Việc nhảy có điều kiện sẽ nhanh hơn việc thiết lập một số điều kiện trong một vi chỉ thị rồi thử nó trong vi chỉ thị tiếp theo. Hầu hết các vi kiến trúc hiện

nay sử dụng chiến lược mà chúng ta vừa nêu dưới dạng này hay dạng khác.

Việc lựa chọn vi chỉ thị tiếp theo được thực hiện bởi khối có ghi nhãn "micro sequencing logic", trong chu kỳ con thứ tư, khi các tín hiệu ra N và Z của ALU đang có tác dụng. Tín hiệu ra của khối này điều khiển bộ dồn kênh Mmux, nó chọn hoặc MPC+1 hoặc là ADDR để cho ra MPC, tín hiệu tại đầu ra MPC sẽ điều khiển việc lấy vi chỉ thị tiếp theo. Người lập vi chương trình có 4 cách lựa chọn vi chỉ thị tiếp theo. Để chỉ ra lựa chọn cần thiết, ta có thể thiết lập trường COND (trường này rộng 2 bit) như sau:

+ 0 = không nhảy, lấy vi chỉ thị tiếp theo từ MPC +1

+ 1 = nhảy tới ADDR nếu N = 1

+ 2 = nhảy tới ADDR nếu Z = 1

+ 3 = nhảy tới ADDR vô điều kiện

Để chiếc máy tính ví dụ của chúng ta giống với máy thật, chúng ta sẽ giả thiết rằng chu kỳ bộ nhớ chính dài hơn chu kỳ của vi chỉ thị. Cụ thể là nếu một vi chỉ thị bắt đầu việc đọc bộ nhớ chính bằng cách đặt RD = 1, thì khi vi chỉ thị tiếp theo được thực hiện cũng vẫn phải có RD = 1. Dữ liệu sẽ có sau khi thực hiện hai vi chỉ thị kể từ lúc bắt đầu đọc. Nếu vi chương trình không có việc hữu ích gì khác để làm tiếp sau một vi chỉ thị cần đọc bộ nhớ thì chúng ta sẽ bị lãng phí thời gian "ngồi chờ" dữ liệu của bộ nhớ. Tương tự như vậy việc ghi bộ nhớ cũng chiếm thời gian bằng hai vi chỉ thị không có truy cập bộ nhớ ngoài.

Sau đây minh họa viết vi chương trình qua một ví dụ lệnh gán đơn giản: cộng AC (thanh ghi số 1) với A (thanh ghi số 10) và chứa kết quả vào AC. Chúng ta có thể viết (lưu ý ở đây dùng số hệ 16 thay cho hệ nhị phân):

A=1h, B=Ah, AMUX=0h, ALU=0h, SH=0h, MBR=0h, RD=0h, WR=0h,
ENC=1h, C=1h, MAR=0h, COND=0h, ADDR=0h;

Tuy nhiên, lập trình vi chương trình như vậy khó kiểm soát, khó nhớ công dụng của từng lệnh và nói chung khó như lập trình bằng lệnh máy nhị phân. Một cách tự nhiên, người ta cũng thay các vi lệnh bằng các lệnh ký tự (ký tự hoá) như là các lệnh hợp ngữ và gọi là vi hợp ngữ (micro assembly language - MAL), tức là mỗi dòng lệnh của hợp ngữ này được dịch ra một dòng vi lệnh gồm các bit điều khiển. Khuôn dạng của mỗi lệnh vi hợp ngữ trông giống một lệnh của Pascal, nhưng cách dịch ra chương trình chạy được lại giống hợp ngữ và mỗi một vi xử lý có một tập lệnh vi hợp ngữ cũng như một bộ dịch riêng đặc trưng. Vi hợp ngữ còn được gọi là VHDL (very hard designed language). Việc ký tự

hoá bắt đầu từ việc đặt tên bằng ký tự gợi nhớ cho các thanh ghi. Ví dụ, thanh ghi số 0 gọi là thanh ghi PC dùng để đếm lệnh, nội dung của nó tăng lên 1 sau khi một lệnh máy nhị phân được đưa vào vi xử lý. Tương tự, thanh ghi 1 là thanh ghi tích luỹ AC thường dùng để chứa kết quả sau khi thực hiện lệnh. Thanh ghi 2 là thanh ghi con trỏ ngăn xếp SP luôn chỉ đến đỉnh ngăn xếp. Thanh ghi 3 là thanh ghi lệnh IR dùng để chứa lệnh máy nhị phân vừa được lấy vào vi xử lý, chờ được giải mã ra vi chương trình. Thanh ghi 4 là TIR (temporary instruction register) thường dùng để chứa tạm kết quả xử lý lệnh máy nhị phân và hỗ trợ IR trong quá trình giải mã lệnh. Các thanh ghi chứa hằng số 0, +1 và -1 rất cần cho các phép đổi ra mã bù và ngược lại, cũng như tăng PC lên 1..., đó chính là tên của thanh ghi 5, 6, 7. Các thanh ghi 8, 9 có tên là AMASK (chứa số nhị phân 0000.1111.1111.1111=0111h) và SMASK (0011h) tức là các thanh ghi mặt nạ dùng để tách trường địa chỉ khỏi mã lệnh... Sáu thanh ghi còn lại đặt tên là A, B, C, D, E, F tuỳ ý sử dụng. Đến đây chúng ta có thể giới thiệu một vài lệnh của vi hợp ngữ.

- Lệnh đọc một ô nhớ địa chỉ chứa trong thanh ghi PC, được viết là : mar:=pc; rd; và được dịch thành vi lệnh:

A=0h, B=0h, AMUX=0h, ALU=2h, SH=0h, MBR=0h, RD=1h, WR=0h, ENC=0h, C=0h, MAR=1h, COND=0h, ADDR=0h;

- Lệnh nạp nội dung MBR vào thanh ghi lệnh IR, được viết là : ir:=mbr; và được dịch thành vi lệnh:

A=0h, B=0h, AMUX=1h, ALU=2h, SH=0h, MBR=0h, RD=0h, WR=0h, ENC=1h, C=3h, MAR=0h, COND=0h, ADDR=0h;

- Lệnh nhân logic nội dung của thanh ghi IR và thanh ghi AMASK nhằm bỏ đi 4 bit đầu tiên chứa mã lệnh, sau đó nhảy về dòng vi lệnh số 1, được viết như sau ac :=band(ir, mask); goto 1; trong đó band là nhân theo đại số Boole và được dịch thành vi lệnh:

A=3h, B=8h, AMUX=0h, ALU=1h, SH=0h, MBR=0h, RD=0h, WR=0h, ENC=1h, C=1h, MAR=0h, COND=3h, ADDR=1h;

- Lệnh dịch trái nội dung thanh ghi IR đi 4 lần, sau khi dịch thì nạp vào thanh ghi TIR để bảo toàn nội dung thanh ghi IR cho lần dùng sau, nhằm kiểm tra bit trọng số lớn thứ 5 ở thanh ghi IR, sau khi dịch là bit trọng số lớn nhất (bit dấu) ở TIR có bằng 1 hay không, nếu bằng 1 thì nhảy đến dòng lệnh 69, sẽ được viết như sau : tir:=lshift(ir+ir); if n then goto 69; lệnh lshift là dịch trái, còn cộng

ir với ir là nhân đôi, thực chất cũng là dịch trái một lần nữa. Lệnh này được dịch thành:

A=3h, B=3h, AMUX=0h, ALU=0h, SH=2h, MBR=0h, RD=0h, WR=0h, ENC=1h, C=4h, MAR=0, COND=1h, ADDR=45h; Lưu ý 45h =69

Qua các ví dụ trên, chúng ta hình dung được thế nào là vi hợp ngữ và bộ dịch phải chứa những vi lệnh tương ứng như thế nào để có thể thực hiện việc thông dịch.

Nhân đây, cần làm rõ việc thông dịch lệnh máy nhị phân ra vi chương trình gồm một số vi lệnh như đã đề cập ở phần tổng quan về các mức máy tính. Lệnh máy nhị phân có thể thông dịch ra vi chương trình bằng hai cách. Cách thứ nhất là viết một đoạn vi chương trình để giải mã các lệnh máy, trong đó chủ yếu dùng lệnh nhân logic, dịch, nhảy có điều kiện để xác định các bit trong trường mã lệnh gồm mấy số 0, mấy số 1 và nằm ở các vị trí nào, từ đó nhận dạng được lệnh máy nhị phân và chỉ đến đoạn vi chương trình tương ứng. Cách thứ hai nhanh hơn, nên hay được dùng, mặc dù có tốn bộ nhớ. Cách này chỉ cần dịch IR sang phải một số lần sao cho chỉ còn trường mã lệnh trong IR (ví dụ còn 7 bit) và nạp vào MPC, tuỳ theo giá trị thập phân của 7 bít này, có 128 cách rẽ nhánh tới 128 đoạn vi chương trình.

Cuối cùng, qua những phân tích trên, rõ ràng bộ dịch cho ngôn ngữ mức trên ra ngôn ngữ mức dưới phải viết bằng ngôn ngữ mức dưới.

V. MỨC HỆ ĐIỀU HÀNH

Vai trò và nhiệm vụ của hệ điều hành là tập hợp các công cụ phần mềm, cho phép sử dụng hữu hiệu phần cứng cho các mục đích khác nhau, quản lý máy tính nói chung, như là: quản lý bộ nhớ, tạo bộ nhớ ảo, quản lý vào ra (I/O), quản lý và khởi động máy tính, đưa máy tính về trạng thái sẵn sàng thực hiện lệnh, tạo môi trường cho các lệnh này chạy được. Đặc điểm quan trọng của hệ điều hành là chiến lược bộ nhớ. Để hiểu hoạt động của hệ điều hành, tiến trình khởi động của hệ điều hành DOS được chọn làm ví dụ minh họa dưới đây. Vấn đề hệ điều hành quản lý máy tính như thế nào sẽ trình bày ở các chương sau.

Trước những năm 90 hệ điều hành DOS được sử dụng rộng rãi trên các máy tính cá nhân. Trên máy IBM-PC có PC-DOS là một hệ điều hành của hãng IBM. Trên máy tương thích với IBM-PC thường có MS-DOS, là một hệ điều hành của hãng Microsoft, nó hoàn toàn tương thích với PC-DOS về phương diện sử dụng.

Thực tế một máy tính có thể được cài đặt và hoạt động với một số hệ điều

hành khác nhau, ví dụ DOS, WINDOWS, CP-M86, XENIX, VENIX... Trong số đó DOS là một hệ điều hành từng được sử dụng sớm và hết sức rộng rãi.

Hệ điều hành MS-DOS bao gồm 6 thành phần cơ bản:

- ROM-BIOS
- Record khởi động
- IO.SYS
- MSDOS.SYS.
- COMMAND.COM.
- Các lệnh ngoại trú

Tên của thành phần thứ ba và thứ tư nêu trên thay đổi tùy thuộc vào nhà sản xuất là IBM hay Microsoft.

1. ROM-BIO

SROM.BIOS là một thành phần nằm trong ROM nói chung. Chip ROM trên mainboard của IBM-PC có thể là ROM kiểu mặt nạ hay EEPROM, phần mềm ghi trong ROM được coi là một trong các thành phần cơ bản của hệ điều hành MS-DOS. ROM chứa các chương trình phục vụ các chức năng sơ đẳng nhất của hệ điều hành và của hệ thống máy tính. Phần lớn nội dung của ROM là các chương trình, ngoài ra còn có một số bảng số liệu về hệ thống.

Việc ghi các chương trình phục vụ cơ sở, thường dùng vào trong ROM có lợi ích rất lớn như sau:

- Không phải nạp từ đĩa mỗi khi khởi động hệ điều hành, nhờ đó có thể sử dụng các chương trình này ngay từ đầu.

- Các chương trình trong ROM được dùng làm cơ sở để xây dựng các chương trình khác, kể cả hệ điều hành. Chúng chính là thành phần dùng chung cho mọi phần mềm chạy trên hệ thống máy tính.

Trong họ máy PC/AT và các máy tương thích, ROM có 4 thành phần là:

- ROM khởi động: Chứa chương trình khởi động, làm những công việc đầu tiên cần cho việc khởi động máy. Nó thường bao gồm một số chương trình con khởi động.

- ROM-BIOS: Chứa tập các chương trình ngôn ngữ máy, thực hiện các thao tác được coi là sơ đẳng nhất của hệ thống máy tính, phục vụ cho hoạt động liên tục của máy tính.

- ROM-BASIC: Làm hạt nhân cho ngôn ngữ lập trình BASIC. Thành phần này thực sự chỉ có trong thế hệ máy đầu tiên.

- ROM mở rộng: Gồm các chương trình thêm vào cho ROM chính, khi người ta thêm vào hệ thống máy tính một thiết bị tuỳ chọn nào đấy.

Phần bộ nhớ cao nhất dành cho ROM, bắt đầu từ đoạn F000h, các máy PC khác nhau sử dụng lượng bộ nhớ khác nhau trong vùng 64K này, PC nguyên thủy dùng 40K, AT dùng cả 64K.

1.1. ROM khởi động (start-up ROM)

Khi bật máy, một chương trình có tên POST (power on self test: tự kiểm tra khi bật máy) bắt đầu chạy, nó thực hiện các chức năng chính sau đây: giám sát quá trình khởi động máy, kiểm tra nhanh độ tin cậy của toàn bộ hệ thống máy tính, kể cả các chương trình trong ROM để khẳng định mọi thứ có thể làm việc được. POST là bước đầu tiên quan trọng nhất để xác định chắc chắn máy vi tính đã sẵn sàng làm việc. Các chương trình con trong POST đều rất ngắn.

a. Kiểm tra bộ nhớ

POST thường kiểm tra bộ nhớ theo khối, với mỗi khối POST chỉ kiểm tra một số byte đầu, nếu các byte được kiểm tra này bình thường thì POST coi cả khối là bình thường. Việc kiểm tra bộ nhớ cần nhiều thời gian vì bộ nhớ thường lớn.

Nếu bộ nhớ bị hỏng ở vùng địa chỉ thấp, nơi bộ vi xử lý cần sử dụng thì POST thường thông báo bằng các tiếng "bip bip" ở loa.

Nếu bộ nhớ bị hỏng ở vùng địa chỉ cao hơn thì POST thường thông báo ra màn hình địa chỉ của ô nhớ hoặc đoạn bộ nhớ bị hỏng, sau đó dừng lại (treo máy).

Trong khi kiểm tra bộ nhớ, POST thường thông báo trên màn hình dung lượng nhớ mà nó đã kiểm tra và thấy là bình thường.

Khi đã kiểm tra xong toàn bộ bộ nhớ và không thấy có vấn đề gì, POST gửi tín hiệu ra loa, chúng ta sẽ nghe thấy một tiếng "bíp".

b. Khởi động (reset) các thiết bị ngoại vi chuẩn được nối với máy

Đó là các thiết bị như bàn phím, màn hình, máy in, ổ đĩa. Chúng ta có thể nhận thấy điều này ở các thiết bị có đèn báo trạng thái. Thí dụ:

Với thiết bị bàn phím, các đèn báo trạng thái Caps, Lock, Scroll, Num Lock sẽ loé sáng khi POST gửi tín hiệu reset tới.

Với màn hình chúng ta không nhận thấy vì POST gửi tín hiệu reset tới chip điều khiển màn hình - CRTC (chip này nằm trên card điều khiển màn hình).

Với máy in, POST gửi tín hiệu reset tới chip điều khiển giao diện với máy in, nếu máy in nối với máy tính và đã bật điện từ trước, chúng ta có thể thấy đèn Ready của máy in nhấp nháy (nếu máy in là loại máy in kim, chúng ta sẽ thấy

đầu in được đưa về đầu dòng...).

Đối với các ổ đĩa, POST gửi tín hiệu reset tới các đơn vị điều khiển chúng, chúng ta có thể thấy các đèn báo trên các ổ đĩa mềm và đèn báo của ổ đĩa cứng (HDD Led) lần lượt lóe sáng. Trình tự thông thường là đèn của các ổ A, B, C rồi D loé sáng. Tuy nhiên trong một số máy trình tự này có thể thay đổi, tuỳ thuộc vào việc máy đã được cài đặt (setup) như thế nào.

Chương trình POST thực hiện reset các thiết bị ngoại vi chuẩn nhằm biết được danh sách các thiết bị chuẩn nối với máy cũng như trạng thái của các thiết bị này, bởi vì hầu hết các thiết bị ngoại vi đều có khả năng thông báo trạng thái của nó cho POST khi bị reset.

Danh sách thiết bị mà POST phát hiện được sẽ được nó ghi vào một vùng nhớ định trước, các chương trình chạy trên hệ thống sau này có thể đọc để biết và tất nhiên cũng có thể thay đổi và thậm chí có thể làm "biến mất" một thiết bị nào đó.

c. Tạo bảng các vectơ ngắt

POST đặt các giá trị định sẵn cho các vectơ ngắt, các giá trị định sẵn này hoặc là trỏ tới một chương trình xử lý ngắt chuẩn nằm sẵn trong ROM - BIOS, hoặc là trỏ tới các địa chỉ bộ nhớ mà nội dung sẽ được các chương trình nạp vào sau này cung cấp.

d. Kiểm tra xem có ROM mở rộng không

Khi bổ sung thêm vào hệ thống máy tính một (hay một số) thiết bị ngoại vi tuỳ chọn nào đó, không phải là thiết bị ngoại vi chuẩn, thiết bị này sẽ không được các chương trình phục vụ có sẵn trong ROM-BIOS hỗ trợ, có nghĩa là nó sẽ không làm việc được. Vì vậy các nhà sản xuất thiết bị ngoại vi này thường ghi sẵn chương trình phục vụ thiết bị của mình vào trong một chip ROM, lắp trên card điều khiển bán kèm theo thiết bị. Chip ROM này được gọi là ROM mở rộng, các chương trình phục vụ trong đó bổ sung thêm vào tập các chương trình phục vụ trong ROM-BIOS. Cũng có khi chúng là các chương trình cải tiến, thay thế cho các chương trình đã có trong ROM-BIOS.

Trong họ máy IBM PC người ta đã đưa ra các quy định về ROM mở rộng. Trong đó có các quy định về việc chương trình trong ROM mở rộng phải bắt đầu tại các địa chỉ nào, tên nhà sản xuất ROM mở rộng ghi ở đâu...

Nếu POST phát hiện thấy có ROM mở rộng thì POST sẽ chuyển điều khiển cho nó để nó tự khởi động lấy, sao cho chương trình trong ROM mở rộng sau

này luôn sẵn sàng làm việc. Chương trình trong ROM mở rộng sau khi tự khởi động luôn phải trả lại điều khiển cho chương trình POST để POST tìm các ROM mở rộng khác và làm nốt các công việc còn lại.

e. Gọi chương trình tải Boot - strap

Công việc cuối cùng của chương trình POST là gọi một chương trình con có nhiệm vụ nạp thành phần tiếp theo của hệ điều hành đang nằm trên đĩa vào bộ nhớ, chương trình con này thường được gọi là chương trình tải Boot-strap (mồi khởi động). Đó là một chương trình ngắn trong ROM thực hiện đọc một sector đặc biệt trên đĩa khởi động (bao gồm việc tìm đĩa khởi động theo thứ tự ghi trong cmos) gọi là bản ghi khởi động (boot record), nếu thành công thì sẽ chuyển điều khiển cho chương trình trong bản ghi này, nếu thất bại sẽ đưa ra màn hình thông báo. Nếu chương trình tải Boot-strap đọc thành công bản ghi khởi động thì chương trình ở bản ghi này sẽ được cho chạy, nó sẽ nạp thành phần tiếp theo của hệ điều hành vào bộ nhớ. Quá trình như vậy được lặp lại cho tới khi tất cả các thành phần cần thiết của hệ điều hành được nạp vào bộ nhớ, sau đó hệ điều hành sẽ làm việc.

1.2. ROM - BIOS

ROM-BIOS (basic input output system) là một phần của ROM, chứa các chương trình phục vụ các thao tác vào (input), ra (ouput) sơ đẳng và cơ bản nhất của hệ thống máy tính, nó luôn luôn được sử dụng tích cực trong suốt thời gian máy tính làm việc.

Phần lớn nội dung của ROM-BIOS là các chương trình con xử lý ngắt, các chương trình này điều khiển các thiết bị ngoại vi chuẩn của máy tính như: màn hình, bàn phím, máy in, ổ đĩa, truyền tin không đồng bộ... trong ROM-BIOS còn chứa các chương trình quản lý ngày, giờ...

Các chương trình của ROM-BIOS chỉ dành cho những người lập trình hệ thống chứ không dành cho người sử dụng bình thường, bởi vì chúng rất sơ đẳng, sử dụng không thuận tiện, chẳng hạn không có khả năng xử lý tệp.

1.3. ROM mở rộng

Chứa các chương trình phục vụ các thiết bị ngoại vi bổ sung. Các chương trình này có mức độ sơ đẳng ngang các chương trình trong ROM-BIOS. Chúng là các chương trình bổ sung, mở rộng hoặc sửa đổi các chương trình đã có trong ROM-BIOS.

2. Record khởi động

Đây là sector vật lý đầu tiên trên đĩa, vị trí của nó trên đĩa cứng và đĩa mềm là như nhau:

Mặt đĩa (head/side) = 0

Rãnh/Từ trụ (track/cylinder) = 0

Cung (sector) = 1

2.1. Sector vật lý

Sector vật lý được chỉ ra bằng 3 tham số như tên, các tham số này còn gọi là địa chỉ vật lý hoặc tọa độ vật lý của sector. Trên mỗi rãnh các sector vật lý được đánh số thứ tự bắt đầu từ 1.

2.2. Sector logic

Hệ điều hành DOS có thể tổ chức một ổ đĩa cứng như một số ổ đĩa logic. Trong mỗi ổ đĩa logic, DOS gán cho mỗi sector một số thứ tự, gọi là số thứ tự sector logic hoặc địa chỉ logic, bắt đầu từ số 0, thường viết tắt là Relsec (relative sector number). Như vậy sector logic là các sector được gắn số thứ tự logic trong các ổ đĩa logic. Có thể tính toán để chuyển đổi từ địa chỉ logic sang địa chỉ vật lý và ngược lại.

2.3. Record khởi động (boot record)

Record khởi động của đĩa mềm và của ổ đĩa logic có chức năng giống nhau, đó là tìm và nạp thành phần tiếp theo của hệ điều hành (file IO.SYS nằm trên đĩa) vào bộ nhớ và trao điều khiển cho nó. Việc này thực tế cũng không phải là đơn giản bởi vì khi chương trình trong boot record chạy, nó chỉ có thể sử dụng các lệnh máy và các dịch vụ sơ đẳng trong ROM-BIOS. Bởi vì chưa có các dịch vụ file nên khi cần đọc đĩa, nó phải sử dụng dịch vụ đọc sector theo địa chỉ vật lý của ROM-BIOS, phải thao tác trực tiếp với bảng FAT và bảng thư mục gốc để lần ra dãy liên tiếp các sector vật lý chứa nội dung file IO.SYS, nạp chúng vào bộ nhớ rồi trao điều khiển (các vấn đề về tổ chức thông tin trên đĩa, trong đó có bảng FAT và bảng thư mục gốc sẽ được trình bày ở phần sau).

Nếu chương trình tải Boot-strap của ROM không đọc được secord khởi động trên ổ đĩa A, nó sẽ tìm trên ổ đĩa C, nếu cũng không đọc được thì sẽ khởi động ROM BASIC (các máy hiện nay không có ROM BASIC). Trong các máy IBM PC và tương thích, chương trình tải Boot-strap cho phép lựa chọn nạp hệ điều hành từ đĩa mềm (A:) hay từ đĩa cứng (C:) hay thậm chí từ CD-ROM thông qua khai báo được ghi lại ở RAM CMOS.

Record khởi động là thành phần đầu tiên nằm trên đĩa của hệ điều hành được nạp vào bộ nhớ và trao điều khiển. Trong quá trình máy tính hoạt động, nó có thể bị các chương trình chạy trên máy sửa đổi, thậm chí thay thế. Nếu điều này xảy ra, trong lần khởi động sau chương trình thay thế hoặc đã sửa đổi này sẽ hoạt động theo ý đồ của người đã viết ra nó. Các harker thường làm như vậy, họ copy nội dung trong boot record cất đi đâu đó mà chỉ có họ biết, sau đó ghi vào record khởi động một chương trình virus. Chương trình virus này khi được chạy sẽ tìm cách thường trú (ẩn nấp đâu đó trong bộ nhớ) và thực hiện một số hành động do harker lập trình trước (thí dụ: lây lan, phá hoại...). Sau cùng nó sẽ đọc boot record "nguyên thủy" mà nó biết nơi cất giấu vào bộ nhớ rồi cho chạy, để chính chương trình "nguyên thủy" này nạp hệ điều hành.

2.4. Master boot record (MBR)

Với đĩa cứng, DOS có thể chia một ổ đĩa thành một số phần. Trên mỗi phần do DOS quản lý, nó có thể tổ chức thông tin như trên một ổ đĩa độc lập duy nhất. Các ổ đĩa này được gọi là ổ đĩa logic, mỗi ổ được DOS gắn cho một tên là một chữ cái, bắt đầu từ C, kết thúc là Z (thực tế cách tổ chức của DOS phức tạp hơn thế nhiều, các độc giả muốn tìm hiểu kỹ hơn có thể tham khảo cuốn sách điện tử "Techhelp" chẳng hạn).

Tổ chức thông tin trên ổ đĩa logic giống tổ chức thông tin trên đĩa mềm, trong đó sector đầu tiên được gọi là sector khởi động (boot sector/boot record). Tuy nhiên với đĩa cứng, record khởi động thực tế là master boot record (MBR), nó không thuộc một ổ đĩa logic nào cả, nó chứa chương trình khởi động chủ. Chương trình này nằm gọn trong một sector 512 byte, có nhiệm vụ chính là xác định xem trong các phần khác nhau của ổ đĩa vật lý (mỗi phần thường được gọi là một partition), phần nào đã được đánh dấu là tích cực (active) thì nạp hệ điều hành trên phần đó vào bộ nhớ và trao điều khiển. Việc nạp này thực tế được chia thành nhiều bước, bước đầu tiên là đọc boot sector trong ổ đĩa logic vào bộ nhớ rồi cho chạy chương trình trong đó. Chính chương trình này sẽ nạp thành phần tiếp theo vào bộ nhớ và trao điều khiển....Quá trình nạp hệ điều hành từ bước này trở đi giống quá trình nạp hệ điều hành từ đĩa mềm.

Chương trình khởi động chủ trong MBR có nhiệm vụ xác định xem sẽ nạp hệ điều hành từ phần nào (nói đơn giản hơn là từ ổ đĩa lôgic nào) trên đĩa cứng, trong khi đó mỗi phần có thể đã được cài đặt một hệ điều hành khác nhau, cho nên chương trình khởi động chủ về nguyên tắc không phải là riêng cho một hệ

điều hành cụ thể nào. Nếu chúng ta phân chia đĩa bằng chương trình FDISK của DOS, nó sẽ ghi chương trình khởi động chủ vào MBR, chương trình này không phụ thuộc vào version của DOS. Nếu chúng ta phân chia đĩa bằng các phần mềm khác thì các chương trình này sẽ ghi vào MBR các chương trình khởi động chủ của chúng. Tuy nhiên, nhiệm vụ chính của chương trình khởi động chủ không thay đổi.

3. IO.SYS

IO.SYS là file đầu tiên trên đĩa, được ghi trên các sector liên tiếp. Tên của nó cũng phải là tên file đầu tiên trong thư mục gốc. DOS quy định như vậy nhằm giúp cho chương trình trong boot record dễ dàng tìm, đọc file này ngay khi các chương trình dịch vụ về file của hệ điều hành còn chưa được nạp vào bộ nhớ. File này được DOS đặt thuộc tính chỉ đọc (read only), ẩn (hidden) và hệ thống (system).

Nhiệm vụ của IO.SYS là mở rộng ROM-BIOS, nghĩa là bổ sung, thay thế hoặc sửa đổi các chương trình phục vụ các thao tác vào/ra sơ đẳng đã có trong ROM-BIOS. Các chương trình phục vụ trong IO.SYS có mức độ sơ đẳng tương đương các chương trình trong ROM-BIOS. Vì IO.SYS được lưu trên đĩa nên nó là một phương pháp tiện lợi để thay đổi ROM-BIOS mà không cần thay chip ROM của máy. Bất kỳ khi nào có một thiết bị mới được thêm vào máy tính thì các chương trình hỗ trợ cho nó phải được đưa thêm vào file IO.SYS hoặc phải có các chương trình điều khiển thiết bị được đưa vào ROM mở rộng để tránh cho việc phải thay đổi ROM-BIOS.

Khi IO.SYS được nạp vào bộ nhớ và trao điều khiển, nó sẽ tìm file CONFIG.SYS là file chứa một số thông tin cấu hình của hệ thống và trong đó có chứa các khai báo tên các trình điều khiển thiết bị ngoại vi bổ sung (driver) cần được nạp vào bộ nhớ. Các trình điều khiển này là sự mở rộng, bổ sung hoặc sửa đổi các trình điều khiển trong ROM-BIOS và trong IO.SYS. Tên các trình điều khiển trong file CONFIG.SYS được viết sau khai báo : DEVICE=.

Vì IO.SYS phải tìm đọc file CONFIG. SYS trong khi chưa có các dịch vụ file của DOS, cho nên DOS đòi hỏi CONFIG. SYS phải có trong thư mục gốc, như vậy việc tìm và đọc sẽ dễ dàng hơn. Sau khi tìm đọc và nạp các trình điều khiển được khai báo trong CONFIG.SYS, chương trình trong IO.SYS tìm đọc và nạp file MSDOS.SYS vào vùng kế tiếp nó trong bộ nhớ và trao điều khiển máy cho MSDOS.SYS.

4. MSDOS.SYS

File này được ghi trên các sector liên tiếp sau vùng dành cho file IO.SYS. Tên của nó cũng phải là tên file thứ hai sau tên file IO.SYS trong thư mục gốc nhằm mục đích dễ tìm file này trong khi các chương trình dịch vụ về file của hệ điều hành nằm chính trong file này còn chưa được nạp vào bộ nhớ. File cũng có những thuộc tính như file IO.SYS.

MSDOS.SYS được nạp vào bộ nhớ ngay sau vùng mà IO.SYS nằm. MSDOS.SYS là hạt nhân của hệ điều hành DOS, nó chứa hầu như tất cả các dịch vụ của DOS, trong đó có tất cả các dịch vụ về file. Mỗi chương trình dịch vụ này có thể được các chương trình khác gọi bằng chỉ thị ngắt (INT) và quản lý I/O ở mức trung gian. Tiếp theo MSDOS.SYS sẽ nạp file COMMAND.COM vào thường trú trong bộ nhớ.

5. COMMAND.COM

Thành phần này có tên gọi là bộ thông dịch lệnh, trong đó chứa tất cả các lệnh nội trú (thường trú) của hệ điều hành DOS. Nó thực hiện nhiệm vụ hiển thị thông báo chờ lệnh trên màn hình, đọc và dịch các câu lệnh nhập từ bàn phím để hệ thống máy tính thi hành.

Để giảm kích thước bộ nhớ bị chiếm bởi command.com, người ta tạo ra file sao cho khi được nạp vào bộ nhớ, nó được tách thành 2 phần: thường trú và bán trú.

- Phần thường trú: Được nạp vào bộ nhớ kế tiếp các thành phần đã được nạp vào trước của DOS, sau đó được bảo vệ chống ghi đè. DOS không phân phối cho các chương trình vùng nhớ được nó bảo vệ.

- Phần bán trú: Được nạp vào bộ nhớ ở vùng địa chỉ cao nhất, sát vùng dành cho ROM và không được bảo vệ chống ghi đè. DOS cho phép các chương trình ứng dụng sử dụng vùng nhớ này. Khi chương trình kết thúc, phần thường trú phát hiện ra, nó sẽ đọc file command.com trên đĩa để khôi phục phần bán trú trong bộ nhớ.

Command.com sau khi được nạp vào bộ nhớ và nắm quyền điều khiển máy sẽ tìm file autoexec.bat tại thư mục gốc, nếu thấy thì cho chạy các chương trình khai báo trong đó.

Việc cuối cùng của command.com là hiển thị thông báo chờ lệnh trên màn hình. Quá trình khởi động kết thúc, máy sẵn sàng nhận lệnh và thi hành.

VI. MỨC HỢP NGỮ (ASSEMBLY)

Như đã trình bày ở phần tổng quan, hợp ngữ dùng các ký hiệu gợi nhớ (mnemonic) để mã hoá lệnh máy và vì thế giúp cho người lập trình làm việc thuận tiện và dễ dàng hơn so với việc viết trực tiếp các lệnh máy. Các chương trình dịch hợp ngữ (assembler) có nhiệm vụ biến đổi các từ gợi nhớ thành các lệnh máy tương ứng 1- 1, sau đó hợp với nhau thành một bản dịch chạy được và cũng chính vì thế mà gọi là hợp ngữ. Hợp ngữ gần gũi nhất với ngôn ngữ máy nên còn được gọi là ngôn ngữ bậc thấp. Hợp ngữ khác với ngôn ngữ bậc cao ở chỗ: các ngôn ngữ bậc cao như Fortran, Pascal, C++.v.v... gần gũi với ngôn ngữ tự nhiên và ngôn ngữ toán, do vậy chúng được sử dụng dễ dàng hơn. Tuy nhiên khác với hợp ngữ, chúng không phản ánh những tiện ích của phần cứng máy vi tính.

Dưới đây là vài ví dụ về một số lệnh hợp ngữ trong bộ lệnh của vi xử lý 8086:

- MOVE AX,15h; nạp số 15h vào thanh ghi AX
- MOVE AX, BX; nạp nội dung của BX vào AX
- OUT 70h, AL; xuất ra cổng 70h (thiết bị ngoại vi 70h) giá trị của thanh ghi AL
- IN AX,60h ; nạp AX giá trị từ cổng 60h (thiết bị ngoại vi 60h)
- MUL BX ; nhân BX với AX, tích số được lưu giữ trong DX: AX
  ; (byte cao trong DX, thấp trong AX)
- Đưa số trong AL ra cổng 70h trong 16 lần :

```
MOV CX,10h ; nạp số 16 vào CX
start
OUT 70h,AL ; đưa số trong AL ra cổng 70h
LOOP start ; lặp lại 16 lần cho đến khi CX =0
```

- MOVE AH,01 ; nạp hàm khởi động máy in của hàm con 17h vào AH
  INT 17h ; gọi hàm con 17h (hàm ngắt 17h) phục vụ máy in

Ví dụ sau là sự so sánh giữa lệnh xuất số 89h ra cổng vào/ra có địa chỉ 303h trong ba ngôn ngữ:

Ngôn ngữ máy:	• 10111010	00000011	00000011	10110000	10001001
	BAh	03h	03h	B0h	89h

Hợp ngữ:
```
MOV  DX, 0303h    ; thanh ghi DX giữ địa chỉ cổng 303h
MOV AL, 89h       ; nạp số 89h vào AL
OUT DX, AL        ; ra số 89h ở cổng 303h
```

Pascal:
```
PORT [$303] : = $89
```

Một chương trình viết bằng hợp ngữ cho vi xử lý 8086 có mẫu chung như sau:

.MODEL SMALL; Kiểu bộ nhớ (còn kiểu MEDIUM, LARGE...).

.STACK 100H; Khai báo kích thước ngăn xếp 100 byte-kích thước hợp lý.

.DATA ; Các định nghĩa số liệu ở dưới đây (biến, hằng).

.CODE; Khai báo đoạn mã lệnh, không nhất thiết có tên khi bộ nhớ small

MAIN PROC ; Tên thủ tục là main, proc đánh dấu bắt đầu.

; Các lệnh ở đây.

MAIN ENDP; Kết thúc thủ tục main.

; Các thủ tục ở đây

END MAIN ; Dẫn hướng biên dịch chỉ chương trình kết thúc

Để dịch chương trình hợp ngữ, có thể dùng chương trình biên dịch MASM hoặc TASM để dịch chương trình gốc hợp ngữ thành tệp có đuôi . obj là bản dịch chứa mã máy của các lệnh trong chương trình gốc. Sau đó dùng chương trình liên kết.

LINK để liên kết một hay nhiều tệp đích và thư viện thành chương trình chạy được có đuôi .EXE.

Bài tập chương 3

1. Lập trình vi chương trình giải hàm sau : X or Y, biết các giá trị X,Y lần lượt nằm trong các thanh ghi số 0 và1. Kết quả lưu trong thanh ghi 15.

2. Lập trình vi chương trình giải hàm sau : (X+ Y) or (X and Y), biết các giá trị X,Y lần lượt nằm trong các thanh ghi 10, 12 . Kết quả lưu vào thanh ghi số 1.

3. So sánh vi lệnh và lệnh máy thông thường.

Chương 4

VI XỬ LÝ 8086

Mục tiêu: Như đã trình bày ở chương 2, mục 3, vi xử lý 8086 có cấu trúc điển hình cho dòng vi xử lý không chỉ của Intel mà còn của Cyrix và AMD, do vậy việc nghiên cứu kỹ lưỡng hơn vi xử lý này rất cần thiết. Cấu trúc chung của vi xử lý, tổ chức của máy PC đã được nghiên cứu tổng quan ở chương 2. Trong chương này sẽ trình bày cho sinh viên vấn đề này sâu hơn và bổ sung một số vấn đề khác có liên quan, đặc biệt là cách thức hệ điều hành dùng để quản lý bộ nhớ của vi xử lý 8086.

I. CÁC THANH GHI CỦA HỌ 80 X 86

Thanh ghi (register) thực ra là một bộ nhớ được chế tạo ngay trong CPU. Vì tốc độ truy cập các thanh ghi nhanh hơn so với bộ nhớ chính RAM nên nó được dùng để lưu trữ các dữ liệu tạm thời cho các quá trình tính toán, xử lý của CPU. Trong họ 80x86 : bề rộng của các thanh ghi là 16 bit với 8088/86/286 và là 32 bit với 386/486. Hình 4.01 liệt kê tất cả các thanh ghi trong chế độ thực (real mode). Bắt đầu từ 80286 có thêm nhiều thanh ghi bổ sung cho việc quản lý bộ nhớ và chúng chỉ có ý nghĩa trong chế độ bảo vệ (protected mode).

AX (AH, AL) : accumulator (thanh ghi tích luỹ)

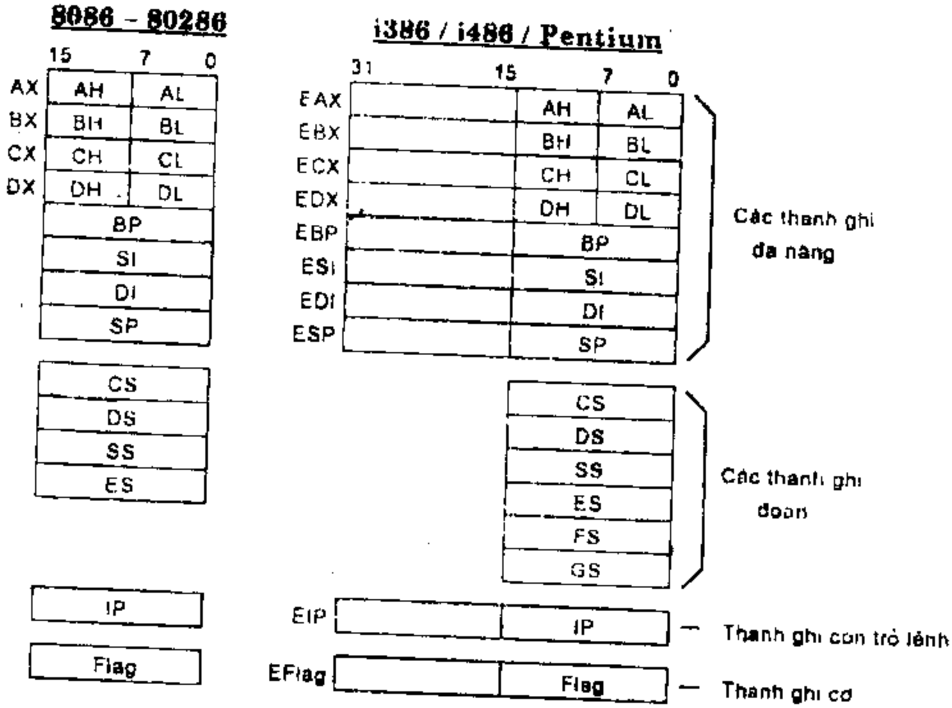
BX (BH, BL) : base register (thanh ghi cơ sở)

CX (CH, CL) : count register (thanh ghi đếm)

DX (DH, DL) : data register (thanh ghi dữ liệu)

BP : base pointer (thanh ghi con trỏ cơ sở)

SI : source index (thanh ghi chỉ số nguồn)



Hình 4-1

DI : destination index (thanh ghi chỉ số đích)

SP : stack pointer (thanh ghi con trỏ ngăn xếp)

CS : code segment (thanh ghi đoạn mã)

DS : data segment (thanh ghi đoạn dữ liệu)

SS : stack segment (thanh ghi đoạn ngăn xếp)

ES : extra segment (thanh ghi đoạn phụ)

IP : instruction pointer (thanh ghi con trỏ lệnh)

F : flag (thanh ghi cờ)

Vi xử lý 8086 có 14 thanh ghi sau : 4 thanh ghi dữ liệu (đa năng hoặc công dụng chung) gồm AX, BX, CX, DX. 5 thanh ghi con trỏ, chỉ số IP, BP. SP, DI, SI. 4 thanh ghi đoạn (segment) CS, DS, BS, ES và một thanh ghi cờ (flag).

Các thanh ghi được truy nhập với từ dài 16 bit (2byte). Riêng với các thanh ghi dữ liệu, có thể được sử dụng như 2 thanh ghi độc lập 8 bit. Để phân biệt

người ta đặt sau tên thanh ghi các chữ H (high) hoặc L (low). Thí dụ AH là thanh ghi tích lũy byte cao, AL là thanh ghi tích lũy byte thấp.

1. Các thanh ghi dữ liệu

1.1. Thanh ghi chứa AX (accumulator register): Thường được sử dụng để chứa các kết quả của các phép tính số học, logic và chuyển dữ liệu, vào ra. Một toán hạng của phép tính nhân, chia được chứa trong AX hoặc AL... Thanh ghi AX giống thanh ghi AC mô tả trong phần vi chương trình.

1.2. Thanh ghi cơ sở BX (base register): Còn được dùng để chỉ địa chỉ cơ sở (đáy) của đoạn nhớ trong bộ nhớ.

1.3. Thanh ghi đếm CX (counter register): Thường dùng để khai báo số lần một thao tác nào đó phải thực hiện trong các vòng lặp, phép dịch, phép quay... giá trị của CX giảm đi với một tác động.

1.4. Thanh ghi số liệu DX (data register): Thường dùng để lưu trữ số liệu dùng làm thông số chuyển giao cho một thường trình. DX là thanh ghi duy nhất dùng cho việc truy nhập các địa chỉ vào/ra trên 255 (FFh) và trong các thao tác nhân, chia.

2. Các thanh ghi đoạn: Bộ nhớ của PC được chia thành các đoạn logic. Người ta sử dụng một thanh ghi 16 bit để chứa và chỉ địa chỉ đoạn này, thành ra có khả năng có 216 = 64K đoạn nhớ. Đồng thời dùng một thanh ghi 16 bit để chứa và chỉ địa chỉ của từng ô nhớ (1B) trong một đoạn, thành ra một đoạn nhớ tối đa có dung lượng 216 = 64KB. Địa chỉ của đoạn hay địa chỉ cơ sở là địa chỉ bắt đầu thấp nhất của đoạn nhớ chứa trong các thanh ghi đoạn so với địa chỉ 0 của bộ nhớ. Còn địa chỉ của ô nhớ trong đoạn nhớ hay còn gọi là địa chỉ tương đối (offset) chứa trong các thanh ghi con trỏ, chỉ số và là địa chỉ của ô nhớ đó so với địa chỉ đoạn. Một chương trình điển hình gồm các lệnh và dữ liệu, ngoài ra còn một cấu trúc dữ liệu khác gọi là ngăn xếp (stack). Mã, dữ liệu và dữ liệu ngăn xếp được nạp vào các đoạn nhớ khác nhau. Do đó có các thanh ghi đoạn sau:

2.1. Thanh ghi đoạn mã CS (code segment register): Dùng để chứa địa chỉ đoạn của đoạn mã lệnh hiện hành trong bộ nhớ.

2.2. Thanh ghi đoạn dữ liệu DS (data segment register): Dùng để chứa địa chỉ đoạn của đoạn dữ liệu hiện hành trong bộ nhớ, nghĩa là nơi chứa các biến, các hằng của chương trình.

2.3.Thanh ghi đoạn ngăn xếp SS (stack segment register): Chứa địa chỉ đoạn của đoạn ngăn xếp trong bộ nhớ. Ngăn xếp là một vùng nhớ đặc biệt, hoạt động liên quan tới các chỉ thị gọi chương trình con. Địa chỉ cũng như kích thước của nó do chương trình ấn định Thông thường, mỗi chương trình có một đoạn ngăn xếp để chứa địa chỉ trở về chương trình chính sau khi thực hiện xong chương trình con. Chương trình con đó có thể lại trở thành chương trình chính cho lần rẽ nhánh tiếp có thể xảy ra... Đoạn ngăn xếp (stack segment) là một tập hợp các ô nhớ tổ chức theo kiểu vào sau ra trước LIFO (last-in/first-out). Số liệu nào được ghi vào ngăn xếp sau cùng thì được đọc ra đầu tiên. Tại các thời điểm đọc hay viết chỉ có một ô nhớ được truy xuất, đó là đỉnh ngăn xếp. Thanh ghi con trỏ ngăn xếp SP (stack pointer) chứa địa chỉ tức thời của đỉnh ngăn xếp.

Các giá trị của thanh ghi và từ nhớ có thể được đẩy vào ngăn xếp với lệnh PUSH. Các cờ có thể được đẩy vào ngăn xếp với lệnh PUSHF và các thanh ghi chung là lệnh PUSHA.

Các lệnh POP, POPF, POPA cho phép hồi phục số liệu từ ngăn xếp.

2.4. Thanh ghi đoạn mở rộng ES (extra segment register): Nếu một chương trình cần truy nhập đến đoạn dữ liệu thứ hai thì phải sử dụng thanh ghi ES. Ngoài ra, ES thường dùng để lưu trữ dữ liệu của chuỗi (string)

3. Các thanh ghi con trỏ và chỉ số

3.1. Thanh ghi con trỏ lệnh IP (instruction pointer): Dùng để lưu trữ địa chỉ của lệnh kế tiếp sẽ được chạy trong đoạn chương trình hiện tại. IP giống như bộ đếm chương trình (program counter : thanh ghi PC) mô tả trong phần vi chương trình. Mỗi lần một từ lệnh được đọc vào từ bộ nhớ, vi xử lý sẽ thay đổi giá trị của IP sao cho nó chỉ đến địa chỉ của từ lệnh kế tiếp trong bộ nhớ. Thanh ghi IP chứa địa chỉ tương đối (offset) trong đoạn nhớ mã lệnh, còn thanh ghi CS chứa địa chỉ đoạn của đoạn mã lệnh. Thanh ghi IP không thể tác động trực tiếp bằng các lệnh, do đó trong một lệnh thường không có mặt IP như một toán hạng.

3.2.. Thanh ghi con trỏ ngăn xếp SP (stack pointer register): Chứa địa chỉ đỉnh ngăn xếp. Giá trị trong SP mô tả địa chỉ tương đối (offset) của đỉnh ngăn xếp, so với địa chỉ đoạn đang được lưu trong SS.

3.3. Thanh ghi con trỏ cơ sở BP (base pointer register): Cũng có chức năng như SP, nhưng còn được sử dụng để truy nhập dữ liệu bên trong ngăn xếp và các đoạn khác.

3.4. Hai thanh ghi chỉ số (index register): Được dùng để lưu trữ các địa

chỉ tương đối trong đoạn dữ liệu. Do đó chúng luôn liên quan tới các trị số trong thanh ghi đoạn dữ liệu DS (data segment). Chúng bao gồm thanh ghi chỉ số nguồn SI (source index) và thanh ghi chỉ số đích DI (destination index). Trong khi SI hay dùng với DS thì DI hay dùng với ES.

4. Thanh ghi cờ: Chỉ có chín trong số 16 bit của thanh ghi này được sử dụng. Mỗi cờ hiệu là mỗi bit và có thể được đặt (=1) hay xoá (=0). Trong phần vi chương trình chúng ta chỉ đề cập tới hai cờ là N và Z, dùng để mô tả trạng thái của ALU sau khi thực hiện xong một thao tác. Vi xử lý 80x86 có 9 cờ như sau:

4.1. 6 bit mô tả các trạng thái (cờ trạng thái để mô tả trạng thái của vi xử lý sau mỗi lần thực hiện một lệnh):

- *Cờ nhớ CF (carry):* Được đặt khi có nhớ từ bit cao nhất MSB (most significant bit) trong phép cộng hay có vay vào MSB trong phép trừ. Ví dụ, khi kết quả của phép lấy tổng hai toán hạng 8 bit quá 255 (= 28-1). Cờ này có thể được đặt bởi lệnh STC (set carry), được xóa bởi lệnh CLC (clear carry) hoặc cộng thêm 1 bởi lệnh CMC (complement carry). Cờ CF cũng bị ảnh hưởng bởi lệnh quay hay dịch.

- *Cờ chẵn lẻ PF (parity):* Được đặt nếu kết quả của một phép toán ở phần byte thấp có chứa một số chẵn bít được đặt (=1).

- *Cờ mang phụ AF (auxiliary carry):* Được dùng như cờ CF nhưng dành cho các thao tác với mã BCD. CF được đặt nếu có nhớ từ bit 3 trong phép cộng và có vay vào bit 3 trong phép trừ.

- *Cờ zero ZF (zero):* Được đặt nếu kết quả của phép toán bằng không (= 0)

- *Cờ dấu SF (sign):* Được đặt nếu bit MSB của kết quả bằng 1, nghĩa là kết quả là một số âm nếu chúng ta đang làm việc với số có dấu.

- *Cờ tràn OF (overflow):* Được đặt nếu kết quả của phép toán quá lớn hoặc quá nhỏ và nằm ngoài phạm vi số đã khai báo (khai báo biến).

4.2. 3 bit cờ hiệu điều khiển (thông qua các cờ này người sử dụng có thể điều khiển một số chức năng của vi xử lý):

- *Cờ báo hướng DF (direction):* Xác định hướng của phép toán xử lý chuỗi. Nếu hướng được đặt, chuỗi sẽ được xử lý địa chỉ cao tới thấp và ngược lại. Cờ này có thể được đặt bởi lệnh STD và được xoá bởi lệnh CLD.

- *Cờ ngắt IF (interrup enable):* Nếu được đặt, CPU sẽ chấp nhận một yêu cầu ngắt cứng IRQs đồng thời sẽ phục vụ ngắt và ngược lại. Cờ này có thể được đặt bởi lệnh STI và được xoá bởi lệnh CLI.

- *Cờ bước TF (trap):* Nếu được đặt, CPU sẽ phát ra ngắt số 1 sau mỗi bước chương trình dùng cho việc gỡ rối (debug). Trong trường hợp này, chương trình được chạy từng bước một. Cờ này có thể không được đặt hay xóa trực tiếp bởi lệnh máy mà phải dùng với sự phụ trợ của ngăn xếp.

Dùng lệnh PUSH đẩy cờ vào ngăn xếp để xử lý cờ ở đó; sau đó dùng lệnh POPF để viết cờ đã được xử lý từ ngăn xếp vào thanh ghi cờ.

Dưới dây là vị trí của các cờ trên các thanh ghi của họ 80x86.

15 ... 0

	...	O	D	I	T	S	Z		A		P		C	8086

N	IO P	O	D	I	T	S	Z		A		P		C	80286

Các bit cờ IOP, N, liên quan đến chế độ hoạt động gọi là chế độ bảo vệ trong 80826.

Trong chương trước, phần vi chương trình chúng ta đã đưa ra một ví dụ vi xử lý có 16 thanh ghi, trong khi vi xử lý 8086 lại chỉ có 14 thanh ghi. Thực chất, thanh ghi lệnh IR trong 8086 chính là hàng nhận lệnh trước PQ có dung lượng 6B đã mô tả ở chương 3. Thay cho thanh ghi chứa hằng số, vi xử lý 8086 có thể dùng các thanh đa năng để nạp các hằng số tương ứng trước khi sử dụng, tất nhiên khi làm như vậy tốc độ xử lý sẽ chậm hơn. Ngược lại, vi xử lý 8086 lại dùng hẳn một thanh ghi để chỉ trạng thái của vi xử lý sau mỗi lần thực hiện lệnh, đó là thanh ghi cờ, trong khi vi xử lý ở phần vi chương trình chỉ kiểm soát kết quả ở ALU với 2 bit N (tương đương cờ SF của 8086) và bit Z (tương đương cờ ZF)

II. QUẢN LÝ BỘ NHỚ CỦA 8086

1. Địa chỉ và bản đồ bộ nhớ

Bus địa chỉ của 8086 có 20 đường dây, do vậy nó có thể quản lý (đánh địa chỉ) đến 220=1M ô nhớ. Vì một ô nhớ bằng một byte nên nói cách khác, không gian địa chỉ vật lý của bộ nhớ 8086 là 1 MB.

Mặt khác, ta thấy rằng các thanh ghi của 8086 chỉ dài 16 bit, tức là nếu dùng một thanh ghi để định địa chỉ thì chỉ có thể định được đến 216=64KB. Để giải quyết vấn đề này, hệ điều hành DOS đã dùng hai thanh ghi để định địa chỉ bộ nhớ theo cách thức sau:

Bằng chương trình, không gian địa chỉ vật lý được phân thành các đoạn nhớ (segment) với kích thước cố định là 64KB mỗi đoạn. Đó là một đơn vị logic của bộ nhớ. Mỗi đoạn gồm các ô nhớ liên tiếp nhau, độc lập và có thể định vị tách rời được. Mỗi đoạn đều được phần mềm gán cho một địa chỉ đoạn là địa chỉ ô nhớ bắt đầu thấp nhất trong đoạn đó. Giá trị của các địa chỉ đoạn này đều phải là 16 hay là bội của 16B. Do đó, các đoạn có thể kế cận nhau, tách rời, phủ lấp một phần lên nhau. Do địa chỉ đoạn cách nhau 16B chúng ta có thể tính ra có bao nhiêu đoạn nhớ trong một bộ nhớ. Tuy nhiên trên thực tế, có những đoạn nhớ ứng với một địa chỉ nào đấy không được dùng do thông tin nạp vào đã phủ lấp lên địa chỉ đoạn đó. Ví dụ có một đoạn tin 33B được nạp vào bộ nhớ từ địa chỉ 0, hay nạp trong đoạn 0, rõ ràng đoạn 1 có địa chỉ đoạn từ byte thứ 16 và đoạn 2 có địa chỉ từ byte thứ 32 không dùng được, vì nếu dùng thì thông tin sau sẽ đè lên thông tin trước. Như vậy từ đoạn 3 có địa chỉ đoạn từ byte thứ 48 mới được dùng và do vậy dễ nhận ra rằng lãng phí tối đa bộ nhớ là 15B. Lãng phí này nhỏ hơn nhiều nếu ta qui định địa chỉ đoạn cách nhau 64KB, tức là các đoạn không phủ lấp lên nhau.

Địa chỉ đoạn sẽ chỉ ra đoạn nhớ bắt đầu từ đâu trong bộ nhớ. Bên trong đoạn tồn tại các giá trị độ dịch (offset), còn gọi là địa chỉ tương đối, chính là khoảng cách tính từ địa chỉ đoạn tới các ô nhớ nằm trong đoạn. Một cặp địa chỉ đoạn và địa chỉ tương đối (segmen : offset) được gọi là địa chỉ logic. Nó cho phép định vị chính xác một byte nhớ trong không gian địa chỉ logic. Các thanh ghi đoạn 16 bit từ CS đến ES được dùng để chứa các giá trị segment và các thanh ghi con trỏ, chỉ số có thể chứa các giá trị offset. Cụ thể cặp thanh ghi CS : IP xác định địa chỉ ô nhớ trong đoạn mã lệnh. Tương tự cặp DS : SI cho đoạn dữ liệu. Còn cặp SS : SP cho đoạn ngăn xếp và cặp ES : DI cho đoạn mở rộng.

Căn cứ vào đó, địa chỉ vật lý thực sự của một byte nhớ được tính từ địa chỉ logic như sau:

Địa chỉ vật lý = 16 x segment + offset.

Việc tính toán này được CPU thực hiện bằng cách dịch trái giá trị thanh ghi đoạn đi 4 bit rồi cộng với giá trị offset (vì khi một số nhị phân được dịch trái đi 4 digit, tương đương với việc nhân nó với $2^4 = 16$).

Rõ ràng ứng với 1 địa chỉ vật lý có thể có nhiều địa chỉ logic trong các đoạn khác nhau. Ví dụ, hai địa chỉ logic sau cùng ánh xạ vào một địa chỉ vật lý 130 493:

1F36h : 0A5Dh đ 1F36h x 16 + 0A5Dh = 130 493

1FB1h : 02ADh đ 1FB1h x 16 + 02ADh = 130 493

Tất cả sự trao đổi thông tin giữa CPU và bộ nhớ đều dùng địa chỉ vật lý, trong khi đó các chương trình được viết theo địa chỉ logic. Khi BIU truy xuất bộ nhớ, nó phát ra một địa chỉ vật lý từ các địa chỉ logic của chương trình.

Vì mỗi thanh ghi đoạn dài 16 bit nên có thể định đến 64K = 65.536 đoạn nhớ và số địa chỉ offset trong một đoạn là 64 KB, do vậy vi xử lý 8086 có thể định địa chỉ tới 64KB x 64K = 4GB nhớ logic.

Để trình bày bản đồ bộ nhớ, sẽ rất tiện lợi nếu phân bộ nhớ trước hết thành các đoạn tách rời nhau : đoạn 0 bắt đầu ở địa chỉ 00000h và kết thúc ở 0FFFFh, đoạn tiếp theo bắt đầu từ 10000h đến 1FFFFh... cho đến đoạn thứ 16 bắt đầu từ F0000h đến FFFFFh. Trong mỗi đoạn rời 64KB có thể có 64K/16 = 4K đoạn nhớ hay bằng 1000h đoạn nhớ theo định nghĩa trên, do vậy tại vị trí bắt đầu đoạn rời số 1 tại địa chỉ 10000h là bắt đầu của đoạn thứ 1000h, tại vị trí đoạn rời số 2 tại địa chỉ 20000h là bắt đầu đoạn thứ 2000h... và tại vị trí đoạn rời thứ 16 tại địa chỉ F0000h sẽ bắt đầu đoạn F000h. Trong phạm vi bộ nhớ qui ước 1MB, hệ điều hành DOS dành cho ROM-BIOS vùng nhớ bắt đầu từ đoạn F000h, đoạn A000h đến đoạn trước C000h dành cho bộ nhớ màn hình, từ đoạn C000h đến đoạn trước F000h dành cho ROM-BIOS mở rộng và dự trữ. Một Kbyte địa chỉ thấp nhất dành cho bảng vector ngắt, ngay sau đó là vùng dành cho hệ điều hành. Vùng tiếp theo vùng dành cho hệ điều hành đến đoạn kề với A000h dành cho các chương trình ứng dụng.

2. Các phương pháp định địa chỉ bộ nhớ

Như trên đã nói để định địa chỉ vật lý khi cần truy xuất bộ nhớ, đơn vị BIU trong vi xử lý phải sử dụng giá trị offset và nội dung thanh ghi đoạn. Giá trị offset mà EU tính cho một toán hạng bên trong đoạn nhớ được chọn gọi là địa chỉ hiệu dụng EA (effective address) của toán hạng. Đơn vị thực hiện lệnh có thể tính EA theo thông tin mã hóa trong byte thứ hai của lệnh. EA được tính dựa trên nội dung của thanh ghi cơ sở, thanh ghi chỉ số và độ dịch chuyển. Độ dịch chuyển (displacement) là một số 8 bit hay 16 bit chứa trong lệnh, thường ở vị trí tên toán hạng.

Với một đoạn mã lệnh hiện hành thì địa chỉ đoạn đã được xác định trong CS. Địa chỉ offset IP sẽ được tự động cập nhật mỗi khi có một lệnh được thực hiện sao cho nó trỏ đến lệnh tiếp theo.

Với đoạn nhớ ngăn xếp hiện hành, địa chỉ đoạn được xác định trong SS. Việc truy cập các ô nhớ trong đoạn nhớ này phải sử dụng các lệnh PUSH và POP như đã trình bày ở mục thanh ghi đoạn SS.

Chỉ còn lại việc truy xuất đoạn nhớ dữ liệu hiện hành mà địa chỉ đoạn đã xác định trong DS hoặc ES và địa chỉ offset được định theo các cách sau:

- *Định vị trực tiếp (direct addressing):* Toán hạng chứa địa chỉ nằm ngay trong lệnh. Địa chỉ đoạn hiện tại chứa trong thanh ghi DS.

Ví dụ: MOV CX, [1234]; chuyển nội dung ô nhớ có địa chỉ offset bằng 1234h trong đoạn số liệu hiện tại vào thanh ghi CX.

- *Định vị gián tiếp thanh ghi (register indirect):* Lúc này EA nằm ở một trong các thanh ghi BX, BP, SI hoặc DI.

Ví dụ: MOV AX, [SI]; chuyển nội dung của ô nhớ trong đoạn hiện tại có địa chỉ offset là nội dung của thanh ghi SI.

- *Định vị cơ sở (based addressing):* EA là tổng của độ dịch chuyển và nội dung của thanh ghi BX hoặc BP.

Ví dụ: MOV [BX]+displacement, AL; chuyển nội dung của thanh ghi AL vào ô nhớ có địa chỉ offset bằng tổng của nội dung BX với độ dịch chuyển.

- *Định vị chỉ số (indexed addressing):* EA là tổng của độ dịch chuyển và nội dung của thanh ghi SI hoặc DI.

Ví dụ : MOV AL, [SI]+ displacement; chuyển nội dung ô nhớ có địa chỉ offset bằng tổng của nội dung SI với độ dịch chuyển vào thanh ghi AL.

- *Định vị chỉ số và cơ sở:* EA là tổng của nội dung thanh ghi cơ sở, thanh ghi chỉ số và độ dịch chuyển.

Ví dụ: MOV AH, [BX] [SI]+ displacement; chuyển nội dung của ô nhớ có địa chỉ offset bằng tổng của độ dịch chuyển với nội dung của BX và SI vào thanh ghi AH.

- *Định vị chuỗi (string addressing):* Dùng cho các xử lý chuỗi, CPU sẽ tự động sử dụng các thanh ghi SI và DI để chỉ đến byte hay từ kế tiếp. SI được dùng để chỉ đến byte hay từ đầu tiên của chuỗi nguồn (DS : SI), DI dùng cho chuỗi đích (ES : DI).

Ví dụ: MOVSB; chuyển một chuỗi các byte.
MOVSW; chuyển một chuỗi các từ.

III. TRUY NHẬP VẬT LÝ BỘ NHỚ VÀ CÁC THIẾT BỊ NGOẠI VI

Ngoài không gian bộ nhớ, PC còn có không gian vào/ra - I/O (Input/Output), gồm các địa chỉ gọi là cổng hay cảng (port) theo sát nghĩa tiếng Anh. Để truy nhập các cổng vào/ra phải dùng các lệnh IN và OUT và chỉ có thể sử dụng chế độ định vị trực tiếp và chế độ định vị gián tiếp sử dụng thanh ghi DX.

Truy nhập có thể với từng byte hoặc từng từ (word). 8086 định ra 64k (=65536) địa chỉ cổng vào/ra 8 bit hoặc 32k địa chỉ cổng 16 bit. Do vậy các thanh ghi đoạn không được sử dụng với truy nhập vào/ra. Đến nay, theo qui ước các máy tính mới chỉ dùng trong phạm vi 1K (= 1024) cổng cho các thiết bị vào/ra. Tuy nhiên việc truy nhập bộ nhớ hay thiết bị ngoại vi đều được vi xử lý thực hiện trong khoảng thời gian gọi là chu kỳ bus.

1. Chu kỳ bus.

Chu kỳ bus là một quá trình xảy ra trên bus để chuyển tải dữ liệu. Khi CPU hay đơn vị làm chủ bus (bus master) thực hiện truy nhập bộ nhớ hay thiết bị vào/ra, nó phải thực hiện một chu kỳ bus (bus cycle). Một chu kỳ bus được CPU hay đơn vị làm chủ bus thực hiện gồm hai giai đoạn: gửi địa chỉ lên bus và chuyển dữ liệu đến hay đi. Giai đoạn đầu gọi là thời gian địa chỉ (address time) trong đó địa chỉ đích được bộ xử lý gửi đi cùng với các tín hiệu điều khiển để xác định loại chu kỳ bus; giai đoạn sau gọi là thời gian số liệu (data time) trong đó bộ xử lý kiểm tra xem đã có tín hiệu sẵn sàng (READY) phát ra từ đơn vị là đối tượng trao đổi dữ liệu hay chưa để cấp dữ liệu.

Có 4 loại chu kỳ bus cơ bản:

Đọc bộ nhớ (memory read)

Viết bộ nhớ (memory write)

Đọc vào/ra (I/O read)

Viết vào/ra (I/O write)

Mỗi chu kỳ bus bao gồm ít nhất 4 nhịp đồng hồ và gồm 4 quá trình từ T_1 đến T_4. Trong khoảng thời gian T_1, 8086 ra địa chỉ của dữ liệu cần đọc/viết. Dữ liệu này được truyền đến đơn vị xử lý trong khoảng thời gian T_3 và T_4. Hướng truyền dữ liệu trên bus sẽ thay đổi trong khoảng thời gian T_2.

Khi ngoại vi được truy xuất cần có nhiều thời gian để trả lời (thí dụ tốc độ xử lý của thiết bị nhỏ hơn tốc độ của CPU), nó phải trì hoãn việc phát ra tín hiệu READY. Lúc đó CPU không nhận được tín hiệu READY ở ngay sau xung nhịp thứ hai mà phải trải qua một khoảng thời gian nữa để kiểm tra tín hiệu này đã có chưa, nếu có mới bắt đầu trao đổi dữ liệu. Mỗi một khoảng thời gian đó gọi là

chu kỳ đợi (wait cycle) ứng với trạng thái đợi (wait state). Rõ ràng, số trạng thái đợi càng nhiều thì hệ thống xử lý càng chậm. Trên một số máy tính, tham số wait state có thể thay đổi được trong phần SETUP của BIOS khi khởi động máy.

2. Các chân nối của 8086 và các tín hiệu

8086 có thể hoạt động ở một trong hai chế độ MAX hoặc MIN:

- Chế độ MIN: CPU tự phát ra tín hiệu điều khiển cho các bus.

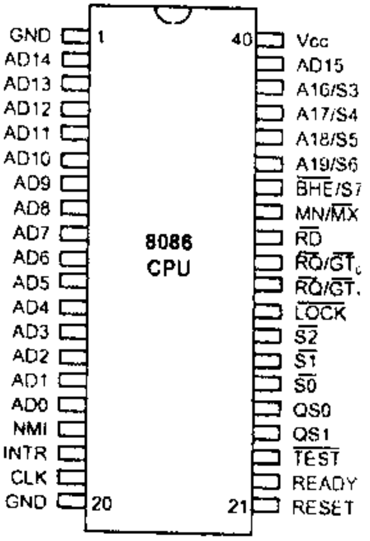
- Chế độ MAX: CPU chỉ phát ra các tín hiệu trạng thái tới một chip điều khiển bus (bus controller), tiếp đó chip này sẽ dịch các tín hiệu trạng thái và phát ra các tín hiệu điều khiển tới các bus. Cách này đảm bảo ít trục trặc hơn trong quá trình đọc số liệu.

Hình 4-2 là sơ đồ chân của 8086 và chức năng của chúng:

2.1. Trước tiên cần đề cập đến các chân (pin out) đa năng của bus A và BHE/S7, các chân này tuỳ theo thời điểm sẽ phát địa chỉ, sau đó là dữ liệu hoặc trạng thái.

AD15-AD0 (chân số 2-16,39): 16 bit số liệu (khi CPU đọc/viết số liệu), vừa là 16 bit địa chỉ thấp (khi CPU định vị bộ nhớ hoặc vào/ra).

A19-A16/S6-S3 (chân số 35-38): 4 bit địa chỉ cao hoặc 4 tín hiệu trạng thái chỉ thị hoạt động hiện tại của CPU. Các trạng thái mô tả trong bảng sau



Hình 4-2

S_4	S_3	Thanh ghi được truy xuất
0	0	ES
0	1	SS
1	0	CS
1	1	DS

S_5 : chỉ thị trạng thái cờ ngắt.

S_6 : luôn bằng 0

BHE/S_7 (chân số 34): Khi vừa khởi động máy tính hay vừa reset, bộ đồng xử lý 8087 sẽ thông qua trạng thái của đường S_7 kiểm tra xem nó có được nối với CPU hay không. Sau đó trong chu kỳ bus, chân này kết hợp với chân địa chỉ A_0 cho các chỉ thị sau:

BHE	A_0	
0	0	một từ đã được truyền qua D_{15}-D_0
0	1	một byte trên D_{15}-D_8 được truy xuất tới một địa chỉ byte lẻ
1	0	một byte trên D_7-D_0 được truy xuất tới địa chỉ byte chẵn
1	1	chưa xác định

RD (chân số 32) 1 : đang đọc bộ nhớ (hoặc vào/ra)

0 : đang viết bộ nhớ (hoặc vào/ra)

Như vậy để đọc một từ bắt đầu từ với địa chỉ chẵn (A_0=0), chỉ cần dùng các đường A_1 đến A_{19} để định địa chỉ và do vậy chỉ cần một chu kỳ bus với BHE=0, A_0=0, nhờ có tín hiệu BHE mà phân biệt được chế độ đọc từng từ hay từng byte. Trong khi đó để đọc một từ với địa chỉ lẻ phải sử dụng đến đường A_0 vì $2^0 = 1$ nhưng A_0 luôn đi cặp với BHE, do vậy cần hai chu kỳ bus để đọc từng byte, nếu BHE = 0, A_0 = 1 thì truy nhập byte lẻ, nếu BHE = 1, A_0 = 0 thì truy nhập byte chẵn. Sau đó BIU sẽ kết hợp hai byte thành một từ ở địa chỉ lẻ. Do đó khi truy nhập các từ số liệu tốt nhất nên đặt chúng ở các địa chỉ chẵn.

2.2. Các chân tín hiệu độc lập dùng cho cả hai chế độ MIN và MAX

READY (chân số 22) : Nếu bộ nhớ (hoặc ngoại vi) cần truy nhập hoàn tất việc chuyển số liệu đến (hoặc đi) chúng cần phát ra tín hiệu READY ở mức tích cực cao tới CPU, chỉ khi ấy CPU mới đọc số liệu vào hoặc tiếp tục ra số liệu.

INTR (chân số 18) : CPU kiểm tra trạng thái chân này sau khi thực hiện xong mỗi lệnh để xem có đòi hỏi ngắt từ phần cứng không; nếu chân này ở mức 1, CPU sẽ nhảy vào dịch vụ ngắt. Kiểm tra này có thể được che (mark) bởi cờ ngắt.

TEST (chân số 23) : lối vào này liên tục được kiểm tra bằng lệnh WAIT
nếu = 0 : CPU tiếp tục chạy chương trình
nếu = 1 : CPU chạy các chu trình giả cho tới khi = 0

NMI (chân số 17) : Nếu thay đổi từ mức 0 đến 1 sẽ gây nên ngắt số 2. Ngắt này không bị che bởi cờ ngắt. Ngắt sẽ được thực hiện ngay sau khi kết thúc lệnh hiện tại. Ví dụ sai số chẵn lẻ của bộ nhớ (memory parity error) sinh ra một NMI.

RESET (chân số 21) : Nếu ở mức 1 trong ít nhất bốn nhịp đồng hồ thì CPU sẽ bỏ nhiệm vụ đang chạy và nhảy vào trạng thái RESET ngay sau khi chân này xuống mức 0.

CLK (chân số 19) : Là lối vào xung nhịp đồng hồ

Vcc (chân số 40) : Nguồn nuôi + 5V cấp cho 8088 qua chân này

GND (chân số 1 và 20) : Được nối đất (thường được nối với OV)

MN/ (chân số 33) : Khi nối với Vcc, CPU hoạt động ở chế độ MIN
khi nối với GND, CPU hoạt động ở chế độ MAX

2.3. Các chân dùng chung cho cả hai chế độ MIN và MAX :

- Các chân số 26, 27, 28 : Trong chế độ MIN chân số 26 là chân tín hiệu DEN, nếu chân này bằng 0 nghĩa là dữ liệu được viết vào bộ đệm dữ liệu và chốt tại đó. Chân số 27 là chân tín hiệu DT/R, nếu bằng 1 nghĩa là vi xử lý đang viết dữ liệu, còn nếu bằng 0 đang đọc dữ liệu. Chân số 28 là chân tín hiệu 0/M, nếu bằng 1 vi xử lý đang truy nhập thiết bị ngoại vi, còn nếu bằng 0 vi xử lý đang truy nhập bộ nhớ.

Trong chế độ MAX các tín hiệu $\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$ lần lượt là các chân từ số 26-28, ở chế độ này chip điều khiển bus 8288 sử dụng ba tín hiệu điều khiển này để phát ra các tín hiệu điều khiển truy xuất bộ nhớ và vào/ra. Tổ hợp các chân có ý nghĩa sau:

0 0 0 : yêu cầu ngắt cứng qua chân INTR được chấp nhận

0 0 1 : đọc vào/ra

0 1 0 : viết vào/ra

0 1 1 : CPU bị treo (halt)

1 0 0 : nạp mã chương trình vào hàng nhận lệnh

1 0 1 : đọc bộ nhớ

1 1 0 : viết bộ nhớ

1 1 1 : trạng thái thụ động

- Các chân số 30, 31: Trong chế độ MIN chân số 31 nhận tín hiệu HOLD, mức tích cực bằng 1 phát từ đơn vị có nhu cầu làm chủ bus nhằm đưa vi xử lý về trạng thái treo và không nắm quyền điều khiển bus nữa. Chân số 31 vi xử lý dùng để phát tín hiệu HLDA xác nhận nhường quyền điều khiển bus.

Trong chế độ MAX hai tín hiệu $\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$ lần lượt là chân số 30, 31, các tín hiệu này phục vụ cho việc chuyển mạch bus cục bộ (local bus) giữa các đơn vị làm chủ bus (bus master). Bus cục bộ là bus giữa các đơn vị xử lý (không phải là bus nối với ngoại vi). Đơn vị làm chủ bus là 8086 hoặc một chip điều khiển nào đó (thí dụ chip DMAC) hiện đang nắm quyền điều khiển bus cục bộ. $\overline{RQ}/\overline{GT_0}$ có mức ưu tiên cao hơn $\overline{RQ}/\overline{GT_1}$. Nếu một đơn vị xử lý khác muốn giành quyền điều khiển bus cục bộ, nó phải đưa ra một tín hiệu yêu cầu qua các chân này tới đơn vị làm chủ bus hiện tại. Nếu có thể chuyển nhượng quyền được, thì sau khi thực hiện xong lệnh hiện tại đơn vị làm chủ bus hiện tại sẽ phát ra tín hiệu ghi nhận (acknowledge) qua các chân này, khi đó đơn vị xử lý kia mới trở thành đơn vị làm chủ bus mới. Cách thức này là cần thiết khi có một vài CPU và các chip vào/ra sử dụng cùng một không gian địa chỉ bộ nhớ vào/ra.

Chân số 29 : Trong chế độ MIN là tín hiệu WR, mức tích cực bằng 0 chỉ thị đang viết bộ nhớ hoặc I/O. Trong chế độ MAX là tín hiệu LOCK, mức tích cực bằng 0 và nếu bằng 0, đơn vị làm chủ bus có thể không nhượng quyền sử dụng bus cục bộ.

Chân số 24, 25 : Trong chế độ MIN chân số 24 phát tín hiệu INTA, mức tích cực bằng 0 thông báo cho phép ngắt (trả lời tín hiệu INT). Chân số 25 phát tín hiệu ALE, nếu bằng 1 nghĩa là đang phát địa chỉ. Trong chế độ MAX các tín hiệu QS_1, QS_0 lần lượt là các chân số 24, 25 để chỉ thị trạng thái của hàng nhận lệnh trước PQ.

0 0 : không hoạt động

0 1 : byte 1 của mã toán trong PQ được xử lý

1 0 : hàng lệnh được xoá

1 1 : byte 2 của mã toán trong PQ được xử lý

Hình 4-3 là sơ đồ chân của chip điều khiển bus 8288, nó là một chip bổ tr cho 8086 và có trách nhiệm phát ra tất cả các tín hiệu cần thiết cho việc điề khiển bus.

Các chân tín hiệu của nó như sau:

MB (chân số 1) : Dùng cho multibus.

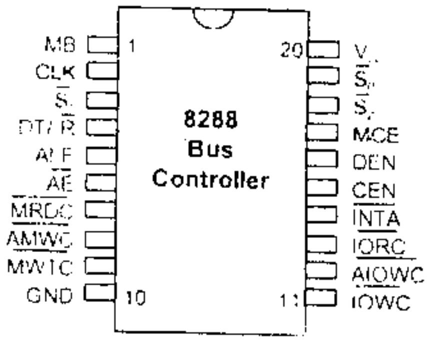
CLK (chân số 2) : Lối vào xung nhịp đồng hồ

S_2, S_1, S_0 (chân số 19, 18, 3) : Lối vào cho tín hiệu từ các chân tương ứng t 8086.

$DT/\overline{R}$ (chân số 4) : 1 : Đang viết số liệu; 0 : Đang đọc số liệu

ALE (chân số 5) : Nếu bằng 1 thì địa chỉ từ CPU gửi tới bộ đệm địa chỉ đượ chốt.

$\overline{MRDC}$ (chân số 7) : Đọc số liệu từ bộ nhớ vào CPU.



Hình 4-3

$\overline{MWTC}$ (chân số 9) : Viết số liệu từ CPU vào bộ nhớ.

$\overline{IOWC}$ (chân số 11) : Viết số liệu từ CPU tới cổng vào/ra.

$\overline{IORC}$ (chân số 13) : Đọc số liệu từ cổng vào/ra vào CPU (chân số 14) : gh nhận yêu cầu ngắt.

INTA (chân số 14) : Nếu bằng 0, ghi nhận yêu cầu ngắt.

DEN (chân số 16) : Nếu bằng 0, số liệu được viết vào bộ đệm số liệu và được chốt ở đó.

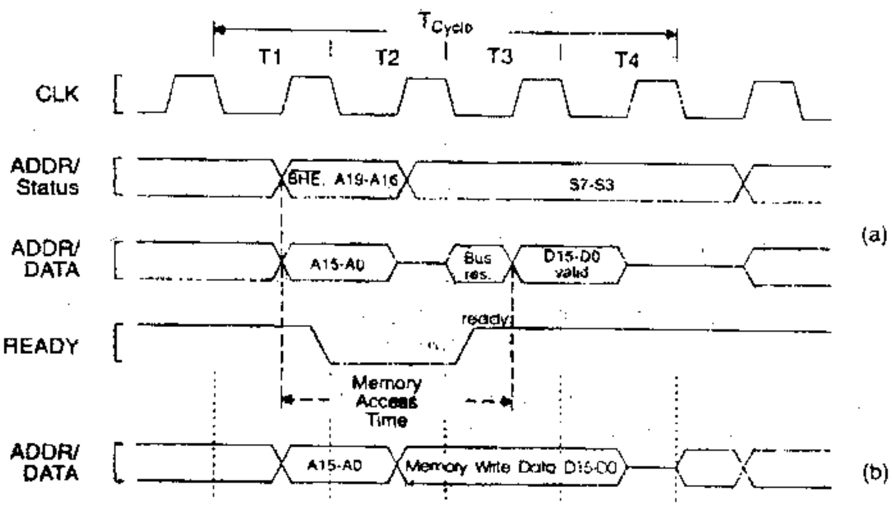
MCE (chân số 17) : Phục vụ cho xử lý ngắt cứng.

3. Truy nhập bộ nhớ chính

3.1. Chu kỳ đọc bộ nhớ

Nhìn trên hình 4-4a ta thấy chu kỳ đọc bộ nhớ bao gồm các quá trình xảy ra như sau:

T_1 : CPU đưa ra tín hiệu địa chỉ trên các chân từ A0 đến A19 và địa chỉ này được chốt vào bộ đệm địa chỉ nhằm cấp cho bus A địa chỉ cho trước liên tục trong thời gian đọc bộ nhớ, mặc dù sau đó vi xử lý chuyển sang phát trạng thái... Tín hiệu chốt địa chỉ vào bộ đệm địa chỉ là ALE. Đồng thời, các tín hiệu điều khiển cũng được phát ra trên bus C (thông qua 8288 trong chế độ MAX) để xác nhận và điều khiển quá trình đọc bộ nhớ. Tín hiệu BHE chỉ thị byte hoặc từ đang được đọc. Tín hiệu READY nhảy xuống mức thấp, nó sẽ chỉ nhảy lên cao một khi bộ nhớ đã cấp xong số liệu. Bộ giải mã loại dồn kênh địa chỉ sẽ định địa chỉ của số liệu cần đọc trong bộ nhớ. Bộ điều khiển bộ nhớ khởi phát xử lý đọc bên trong bộ nhớ chính.



Hình 4-4

T_2 : Đường A_{19}-A_{16} chuyển sang thông tin trạng thái. Các đường A_{15}-A_0 chuyển từ chế độ địa chỉ sang số liệu.

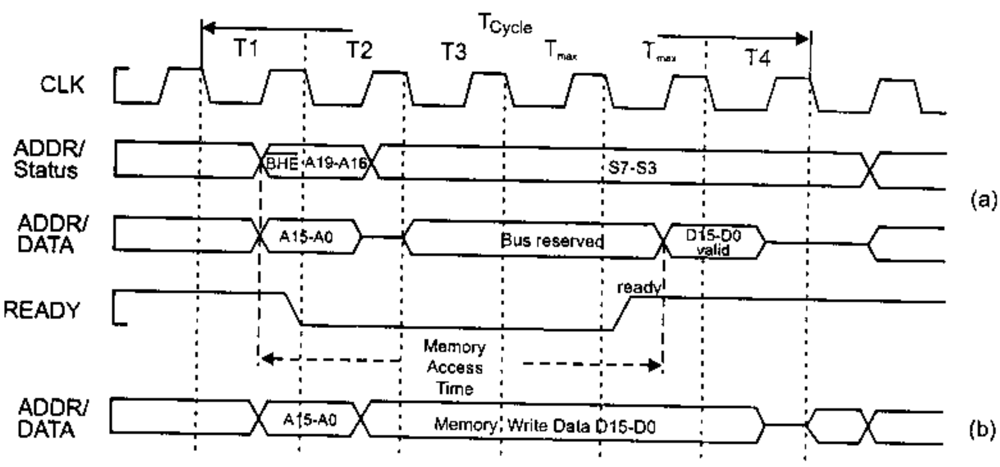
T_3 : Chu trình truyền số liệu bắt đầu. Chừng nào số liệu chưa ổn định trên D_{15}-D_0 thì các tín hiệu trạng thái S_3-S_6 sẽ được xuất ra trên các đường A_{16} đến A_{19}. Khi toàn bộ số liệu được truyền xong, bộ điều khiển nhớ sẽ nâng mức điện thế ở dây READY lên cao. Để đồng bộ với xung nhịp, tín hiệu READY phải được đi qua bộ đồng pha. Như vậy số liệu đã được truyền từ bộ đệm bộ nhớ tới bộ đệm số liệu. CPU lúc này khởi phát việc nhận số liệu từ bộ đệm số liệu.

T_4 : CPU kết thúc việc đọc số liệu vào sau 1/2 chu kỳ nhịp. Lúc này các bộ đệm bị cấm nhưng CPU vẫn liên tục cho ra các tín hiệu trạng thái S_7-S_3. Sau khi kết thúc T_4, bus hệ thống lại một lần nữa trở về trạng thái khởi phát.

3.2. Chu kỳ viết bộ nhớ

Các tín hiệu (hình 4-4b) giống như ở chu kỳ đọc, ngoại trừ tín hiệu ADDR/DATA, chỉ lưu ý hướng của bus địa chỉ/số liệu không cần đổi vì cả địa chỉ và số liệu đều là hướng ra (từ CPU), khác với chu kỳ đọc. Do đó ngay sau khi cấp địa chỉ, CPU có thể phát ra số liệu vào bộ đệm ở xung nhịp đồng hồ trong T_2 không cần một thời gian đợi (bus reserved). Bộ đệm số liệu truyền số liệu tới bộ đệm nhớ, đồng thời bộ điều khiển nhớ sẽ điều khiển bộ nhớ viết số liệu vào trong đó. Tín hiệu READY cũng lên cao để chỉ thị tới CPU khi hoàn tất việc viết số liệu vào bên trong bộ nhớ. CPU kết thúc quá trình viết. Các bộ đệm bị cấm nhưng CPU tiếp tục ra các tín hiệu trạng thái S_7-S_3.

3.3. Truy nhập bộ nhớ chính với các trạng thái đợi



Hình 4-5

Thời gian truy nhập bộ nhớ (access time) là khoảng thời gian kể từ khi tín hiệu địa chỉ xác định tới khi tín hiệu số liệu được phát ra ổn định. Nếu tốc độ truy nhập của chip nhớ quá chậm (hay tốc độ xung đồng hồ quá cao) thì tín hiệu READY cho phép một đáp ứng truy xuất mềm dẻo bằng cách xen vào các trạng thái đợi như hình 4-5 a.b.

Bộ điều khiển nhớ điều khiển các trạng thái đợi bằng việc phát ra tín hiệu READY chậm hơn một hoặc vài nhịp đồng hồ cho tới khi số liệu có sẵn trên bus số liệu (chu kỳ đọc) hoặc số liệu đã được viết vào bộ nhớ (chu kỳ viết). Với một bộ nhớ chính thông thường, số trạng thái đợi được cố định hoặc có thể tuỳ chọn trong RAM-CMOS. Số trạng thái đợi có thể khác nhau khi đọc hay viết và phụ thuộc vào vùng nhớ. Thường chip nhớ có thể đọc số liệu nhanh hơn viết. Quá trình truy nhập bộ nhớ chính trên bản mạch chính thường chạy với số chu kỳ đợi ít hơn trên RAM video.

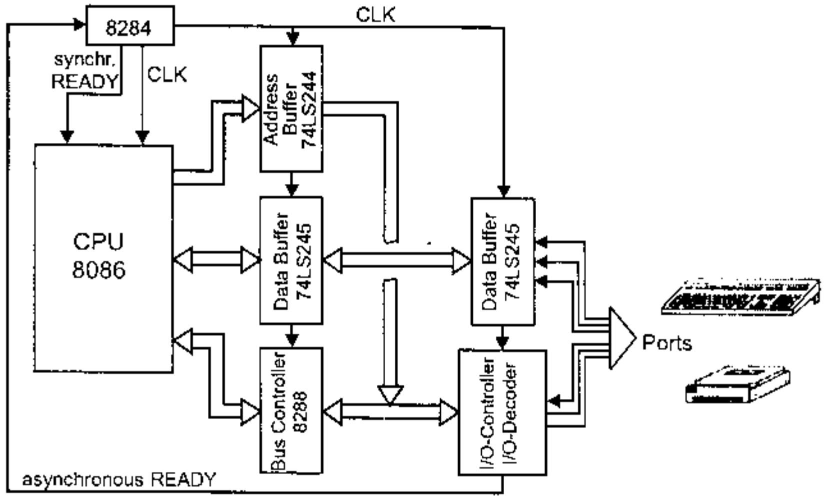
Hình 4-5a là giản đồ thời gian đọc bộ nhớ : Bộ xử lý cung cấp tín hiệu địa chỉ và đợi số liệu từ bộ nhớ chính; bộ nhớ chính có thời gian truy nhập dài nên không thể phát ra số liệu kịp thời và phải trì hoãn việc kích hoạt dây READY cho đến khi số liệu ổn định trên bus số liệu.

Hình 4-5b là giản đồ thời gian viết bộ nhớ : Bộ xử lý ra tín hiệu địa chỉ và ngay sau đó viết số liệu; bộ nhớ chính làm việc không đủ nhanh khi lấy số liệu, do vậy nó cũng phải trì hoãn việc kích hoạt dây READY đến khi số liệu được viết xong vào ô nhớ. Cả hai chu trình bao gồm sáu nhịp đồng hồ, tức là đã có hai trạng thái đợi T được xen vào.

4. Truy nhập các cảng vào/ra.

Khi cần truy nhập các cảng vào/ra, bộ điều khiển bus sẽ làm tích cực các lối ra như IORC (đọc số liệu) hoặc IOWC (viết số liệu). Cách tiến hành truy nhập, chu kỳ bus cũng giống như truy nhập bộ nhớ. Hình 4-6 mô tả các khối giữa CPU và cổng trong truy nhập vào/ra.

CPU phát ra một tín hiệu địa chỉ tới bộ đệm địa chỉ và gửi tín hiệu trạng thái tới bộ điều khiển bus 8288 nếu là chế độ MAX. Nếu số liệu được truyền tới cổng (chế độ ra, OUT), CPU sẽ gửi số liệu tới bộ đệm số liệu trong thời gian T_2. Các tín hiệu điều khiển $\overline{\text{OIRC}}$ và $\overline{\text{IOWC}}$ cho phép mạch logic trên bản mạch chính nhận ra đây là truy nhập vào/ra chứ không phải truy nhập bộ nhớ. Do đó, thay vì cho bộ điều khiển nhớ, lúc này là bộ điều khiển vào/ra sẽ hoạt động. Chỉ lưu ý rằng bộ giải mã địa chỉ chỉ cần 10 đầu vào ứng với các đường A_0 đến A_9 do CPU chỉ sử dụng 1024 địa chỉ cổng. Do thiết bị ngoại vi hoạt động chậm hơn các bộ nhớ chính nên chu kỳ bus cũng được xen vào các trạng thái đợi.



Hình 4-06

IV. CÁC CHIP HỖ TRỢ

Như đã trình bày ở chương 2, chip hỗ trợ hoạt động cho CPU có nhiều, phần này chỉ đi sâu nghiên cứu hai chip quan trọng nhất là hỗ trợ ngắt và DMA.

1. Chíp điều khiển ngắt PIC - 8259A (programmeble interrupt controller)

1.1. Khái niệm chung về ngắt : Ngắt là khả năng dừng chương trình chính đang chạy khi có đòi hỏi ngắt, để thực hiện một chương trình khác gọi là chương trình con xử lý ngắt. Chương trình con này kết thúc bằng lệnh IRET(trở về chỗ ngắt). Gặp lệnh này CPU quay về thực hiện tiếp chương trình chính đúng từ chỗ bị ngắt.

1.2. Phân loại ngắt

a. Ngắt mềm : Là ngắt được gọi bằng một lệnh trong chương trình. Với hợp ngữ đó là lệnh INT và chỉ số ngắt (dùng số hệ 16). Ví dụ lệnh gọi ngắt số 5 được viết là INT5h (in trang màn hình - print screen) . Khi gặp lệnh này, vi xử lý sẽ dừng chương trình chính đang chạy, cất địa chỉ lệnh tiếp theo vào ngăn xếp và lấy chỉ số ngắt nhân với 4 ra địa chỉ của đoạn nhớ 4B chứa địa chỉ của chương trình con phục vụ ngắt (gọi là véc tơ ngắt), truy nhập địa chỉ này để chạy chương trình con phục vụ ngắt có trong ROMBIOS. Chú ý rằng khi xuất hiện một ngoại lệ (exception), một ngắt mềm được sinh ra . Dưới đây là một bảng các ngoại lệ với số ngắt tương ứng:

Interrupt	Mô tả ngoại lệ
0h	division by zero : chia cho không;
1h	single step : chạy chương trình từng bước;
3h	breakpoint : điểm dừng
4h	overflow detection with INTO : phát hiện tràn số với INTO
6h	invalid opcode : mã toán không hợp lệ;
7h	coprocessor not present : không có bộ đồng xử lý

b. Ngắt cứng: Được gọi nhờ tín hiệu điện được sinh ra bởi các chip điện tử trong PC hoặc thiết bị ngoại vi có nhu cầu được phục vụ. Đó là một cơ cấu đơn giản và hiệu quả để CPU phản ứng một cách kịp thời với các yêu cầu ngắt. Thí dụ, ấn hay nhả bàn phím sẽ gây nên ngắt cứng số 9 ký hiệu là IRQ9, chương trình xử lý ngắt sẽ phản ứng bằng cách đưa ký tự được ấn vào vùng nhớ đệm của bàn phím, vào vị trí ngay sau ký tự được ấn lần trước. Nếu vùng đệm đầy, sẽ sinh ra tiếng bíp.

Mỗi một ngắt cứng tương đương với một ngắt mềm xác định. Bảng sau liệt kê danh sách tương đương hai loại ngắt dùng cho một thiết bị nào đấy:

IRQ0	INT8h	Đồng hồ hệ thống
IRQ1	INT9h	Bàn phím
IRQ2	INTAh	8259 tớ
IRQ3	INTBh	COM2
IRQ4	INTCh	COM1
IRQ5	INTDh	LPT2
IRQ6	INTEh	Đĩa mềm
IRQ7	INTFh	LPT1
IRQ8	INT70h	Đồng hồ thời gian thực
IRQ9,10,11	INT71h-73h	Dự trữ
IRQ12	INT74h	Chuột PS/2
IRQ13	INT75h	Bộ đồng xử lý toán học
IRQ14	INT76h	HDD1
IRQ15	INT77h	HDD2 hay CD-ROM

Ngắt cứng được chia làm hai loại là ngắt che được và không che được.

- Ngắt có thể bị che (maskable): Có thể bị cấm (che) bằng lệnh hợp ngữ CL xóa cờ ngắt, khi đó cờ IF=0. Nếu bị che thì mặc dù được gọi, chương trình xử lý ngắt tương ứng cũng không được thực hiện. Lệnh STI (đặt cờ ngắt) cho phép các ngắt bị che trở lại hoạt động. Ký hiệu ngắt này là IRQxx hay IRxx, trong đó xx là chỉ số ngắt.

- Ngắt không che được (non-maskable - NMI): Luôn được thực hiện kể cả khi ngắt này được gọi ngay sau lệnh CLI. Ngắt này liên quan tới các hỏng hóc phần cứng nghiêm trọng (thí dụ, hỏng RAM). Ngắt NMI tương đương ngắt mềm INT2h.

Ngoài ra, ngắt có thể được phân loại theo quan điểm hệ thống : ngắt trong hay ngắt ngoài.

1.3. Sơ đồ khối của chip 8259

Vì có nhiều ngắt đến từ các thành phần khác nhau của hệ thống có thể xuất hiện cùng một lúc, trong khi vi xử lý chỉ có một lối vào (INTR) nhận tín hiệu ngắt nên trước tiên chúng cần phải được chuyển tới một chip điều khiển ngắt PIC (programmable interrupt controller). Mỗi PIC8259 có khả năng đồng thời nhận tám yêu cầu ngắt IRQ (hay IR). PIC gán cho mỗi yêu cầu ngắt một mức ưu tiên, dựa vào chức năng chính của yêu cầu ngắt và chuyển cho CPU yêu cầu ngắt có mức cao nhất trước.

Hình 4-7 là sơ đồ khối bên trong và các chân của PIC 8259.

Các yêu cầu ngắt được xử lý bởi 4 thanh ghi:

Thanh ghi IRR (interrupt request register): Ghi các yêu cầu ngắt từ đầu vào IR0-IR7.

Thanh ghi ISR (interrupt service register): Ghi yêu cầu ngắt đang được phục vụ.

Thanh ghi IMR (interrupt mask register): Thanh ghi che ngắt cho IR0 đến IR7.

Thanh ghi PR (priority register): Thanh ghi xử lý ưu tiên.

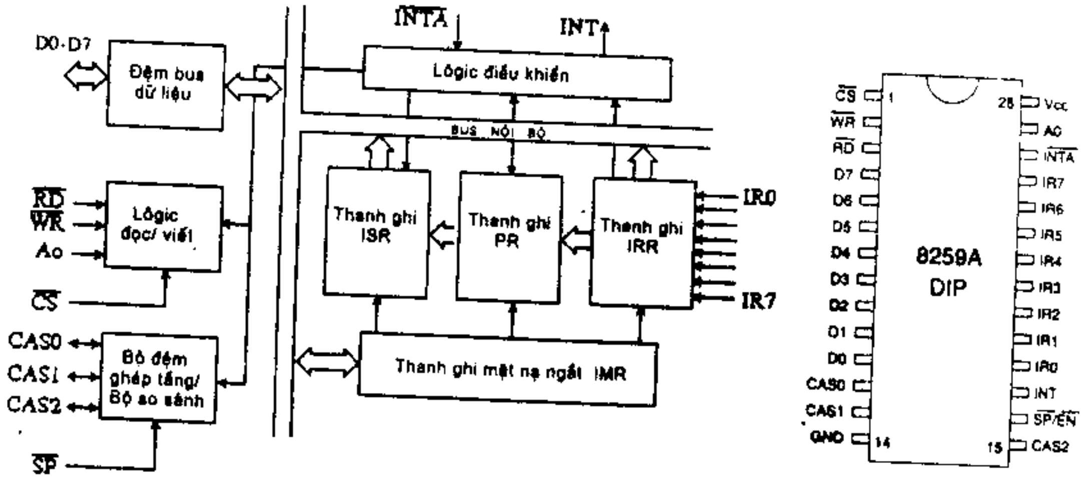
Mạch logic giải quyết ưu tiên PR(priority register) trên cơ sở nội dung của bốn thanh ghi này quyết định yêu cầu ngắt INT gửi đến CPU hay không thông qua logic điều khiển (control logic) của 8259.

Các chân và tín hiệu của PIC - 8259 như sau:

$\overline{CS}$ (chân số 1): Lối vào tín hiệu chọn chip.

$\overline{WR}$ (chân số 2): CPU có thể viết số liệu vào các thanh ghi bên trong của 8259.

$\overline{RD}$ (chân số 3): CPU có thể đọc số liệu từ 8259.



Hình 4-7

D0-D7 (chân số 4-11): bus số liệu hai hướng

CAS0-CAS2 (chân số 12, 13, 15): Các đường nối tầng giữa các PIC. Một PIC chủ có thể chọn một trong 8 PIC tớ qua 3 đường này. Do đó có thể coi đây là một bus địa chỉ cục bộ của PIC.

GND (chân số 14): nối đất (OV)

$\overline{SP}/\overline{EN}$ (chân số 16): Trong chế độ đệm bus dữ liệu, $\overline{EN} = 0$ điều khiển mở thông 8259 với bus hệ thống, lúc này việc định nghĩa vi mạch chủ được thực hiện thông qua từ điều khiển ngắt ICW4 (interrupt control word). Trong chế độ không đệm, nếu $\overline{SP} = 1$ thì 8259 là chủ, ngược lại $\overline{SP} = 0$ thì 8259 là tớ.

INT (chân số 17): nối trực tiếp với đầu vào INTR của CPU.

IRO-IR7 (chân số 18-25): Nối với các đầu ra tín hiệu ngắt của các thiết bị ngoài

$\overline{INTA}$ (chân số 26): Tín hiệu vi xử lý trả lời chấp nhận ngắt.

A0 (chân số 27): Cùng với các tín hiệu phân biệt các lệnh khác nhau từ CPU và cung cấp các thông tin trạng thái.

Vcc (chân số 28) nguồn nuôi +5V

Các PIC - 8259 có thể được ghép tầng (cascade) với nhau. Chúng được phân

thành hai mức: một PIC chủ (master) và từ 1 đến 8 PIC tớ (slave). Khi đó, hệ có thể quản lý được các yêu cầu ngắt của hơn 8 thiết bị ngoại vi (thí dụ như ở máy tính AT). Lúc đó lối ra INT của PIC tớ được nối với lối vào IR (thường dùng IR2) của PIC chủ và các tín hiệu từ CASO đến CAS2 được dùng cho việc trao đổi thông tin giữa chúng. Do vậy với một cặp nối tầng PIC một chủ, một tớ có thể quản lý được 15 kênh yêu cầu ngắt. Trong đó, các ngắt IR0 đến IR7 (trừ IR2 dùng làm lối vào cho PIC tớ) được nối vào PIC chủ, còn IR8 đến IR15 được nối vào PIC tớ.

1.4. Thủ tục ngắt

Thiết bị ngoài gửi yêu cầu ngắt tới 8259, một trong các đường yêu cầu ngắt IRO-IR7 được nâng lên mức cao, bit tương ứng trong IRR được đặt. 8259 gửi tín hiệu INT=1 tới CPU. CPU nhận tín hiệu INT và ra xung /INTA=0 thứ nhất nếu cờ IF được đặt, cất địa chỉ lệnh nơi dừng chương trình vào ngăn xếp, báo cho 8259 biết yêu cầu ngắt đã được chấp nhận. 8259 sẽ hoàn tất các xử lý nội bộ theo thứ tự ưu tiên. Bit ưu tiên cao nhất trong IRR được xóa, bit tương ứng được đặt trong ISR. CPU ra xung $\overline{INTA}$ thứ hai, 8259 đưa con trỏ 8 bit chỉ số ngắt lên bus số liệu. CPU đọc và gọi ngắt. Chạy xong chương trình ngắt, CPU truy nhập ngăn xếp lấy địa chỉ trở về và chạy tiếp chương trình chính vừa bị ngắt (dừng). Trong chế độ tự động, bit trong ISR tự động được reset: nếu không CPU phải ra lệnh EOI tới 8259 khi chạy chương trình ngắt để xóa bit ISR.

1.5. Truy nhập 8259

Chế độ làm việc của 8259 được xác lập bởi các từ điều khiển do phần mềm nạp vào các thanh ghi, cũng như việc đọc các thanh ghi nhờ các lệnh truy nhập thiết bị ngoài (IN,OUT). Có hai loại từ điều khiển : các từ điều khiển khởi động ICW (intialization control word) được nạp vào khi khởi động máy tính và các từ điều khiển hoạt động OCW (operation control word). Các thanh ghi của PIC - 8259 trong PC/XT cũng như PIC - chủ trong máy AT được thâm nhập qua các cảng 20h và 21h và PIC-tớ trong AT được thâm nhập qua cảng A0h và A1h.

Có 4 từ ICW : ICW1, ICW2, ICW3, ICW4, riêng ICW4 chỉ dành cho vi xử lý 16 bit trở lên. Từ ICW tuỳ theo giá trị của các bit điều khiển sẽ xác định tín hiệu ngắt cứng tác động theo sườn xung hay theo mức, khoảng cách giữa các

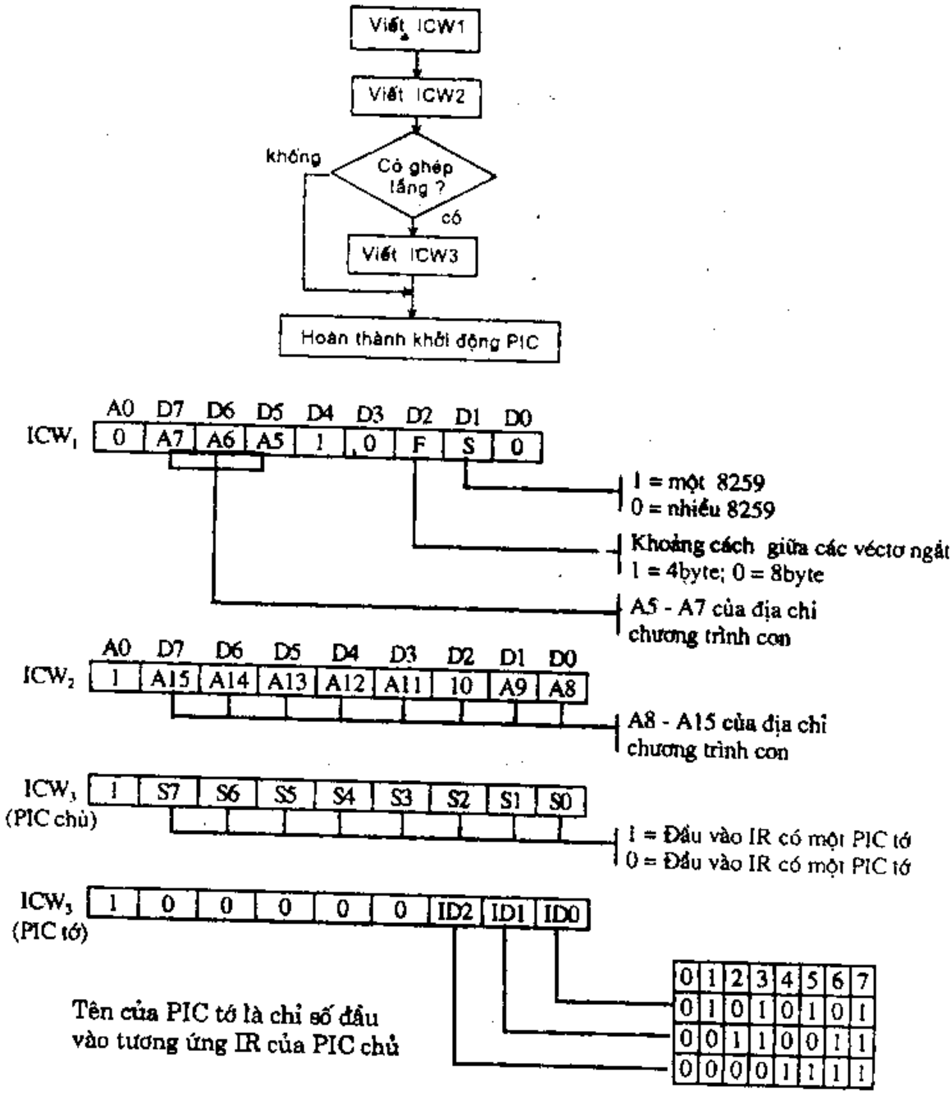
vector ngắt là 4B hay 8B, chỉ dùng một PIC chủ hay không. Từ ICW2 xác định kiểu ngắt, ví dụ 8h nghĩa là IRQ0=INT8h. Từ ICW3 chỉ được nạp vào nếu có dùng PIC tớ và xác định PIC tớ được nối vào PIC chủ ở chân IR số bao nhiêu, ví dụ ICW3=04h nghĩa là nối vào chân IR2.

Ba từ OCW: OCW1, OCW2, OCW3 được nạp vào các thanh ghi tương ứng để điều khiển hoạt động của 8259. Từ OCW1 được nạp vào IMR để che một yêu cầu ngắt nào đấy theo định nghĩa. Từ OCW2 định chế độ ưu tiên ngắt (xem bảng dưới đây). Từ OCW3 chủ yếu dùng thăm dò trạng thái yêu cầu ngắt để quyết định dùng chương trình phục vụ ngắt nào nếu tồn tại nhiều chương trình cho một yêu cầu ngắt.

Chế độ	**Chức năng, ý nghĩa**
Ưu tiên cố định (full nested)	Mỗi đầu ngắt được gắn mức ưu tiên cố định. Mức 0 là cao nhất, mức 7 là thấp nhất.
Ưu tiên vòng tự động(auto rotating)	Ưu tiên đồng đều, mỗi mức sau khi được phục vụ trở thành mức thấp nhất cho đến khi có ngắt tiếp xảy ra.
Ưu tiên vòng theo quy định (specific rotating)	Phần mềm của hệ quy định mức ưu tiên thấp nhất. Thứ tự ưu tiên được lập trên cơ sở quy định này.
Hỏi vòng ngắt (polled)	Phần mềm của hệ đọc thanh ghi trạng thái của 8259, từ đó xác định nguồn ngắt và quyết định phục vụ ngắt.

Chế độ ưu tiên cố định thường được sử dụng. Sau khi khởi động, 8259 xử lý ngắt theo chế độ ưu tiên cố định, không nhận bất cứ một OCW nào. Ở chế độ này IR0 là mức ưu tiên cao nhất và theo thứ tự IR7 có mức ưu tiên thấp nhất.

Hình 4-8 trình bày lưu đồ khởi động tổng quát của 8259 và định dạng của từng từ khởi động ICW.



Hình 4-8

2. Chip truy nhập bộ nhớ trực tiếp DMA-8237 (direct memory access controller)

2.1. Khái niệm DMA

Để trao đổi dữ liệu giữa bộ nhớ và các bộ phận ngoại vi, chúng ta cần phải nạp qua một thanh ghi trong vi xử lý. Quá trình trao đổi được thực hiện nhờ các lệnh hợp ngữ MOV, IN, OUT tốn rất nhiều thời gian. Thường một lệnh chuyển số liệu phải sử dụng một chu kỳ bus kéo dài ít nhất 4 chu kỳ xung nhịp (4T),

chưa kể thời gian giải mã và thực hiện lệnh bên trong vi xử lý (có thể đánh giá cũng mất cỡ thời gian kéo dài 4 chu kỳ xung nhịp - xem chương 3). Như vậy để chuyển một số liệu giữa bộ nhớ và thiết bị ngoại vi qua trung gian và kiểm soát của vi xử lý mất ít nhất 2x8T=16T.

Phương pháp DMA cho phép tách vi xử lý ra khỏi hệ thống bus, khi ấy số liệu có thể chuyển trực tiếp giữa bộ nhớ và thiết bị ngoài. Vì bộ nhớ và thiết bị ngoài vẫn dùng chung một bus D, nên chip DMA, lúc này nắm quyền điều khiển bus, sẽ phải phát tín hiệu đọc (hoặc viết) bộ nhớ nhằm đưa số liệu của ô nhớ cần truy nhập lên bus D, sau đó lại phát tiếp lệnh viết (hay đọc) để nhập số liệu vào thiết bị ngoài. Số lần phát lệnh chính là số từ đã chuyển. Có thể đánh giá phương pháp này mất hai xung nhịp (2T), nhanh hơn so với cách truy nhập gián tiếp qua vi xử lý.

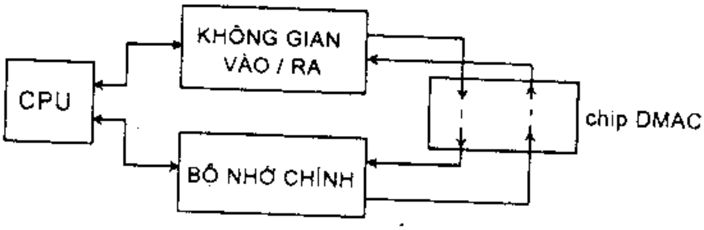
Thông thường để trao đổi số liệu có thể dùng phương pháp ngắt. Phương pháp này đảm bảo thời gian trả lời yêu cầu vào/ra số liệu của ngoại vi là rất ngắn. Nhưng việc phục vụ quá trình vào/ra lại được thực hiện bằng phần mềm, do đó khi truy nhập một số ngoại vi như bộ hiển thị màn hình CRT hay bộ nhớ khối (đĩa cứng, đĩa mềm...) để chuyển một mảng số liệu thì phương pháp ngắt không đủ nhanh để đáp ứng yêu cầu chuyển số liệu rất cao. Lúc này phải dùng giải pháp bằng phần cứng hay DMA nói trên. Chương trình con thực hiện chuyển số liệu giữa ngoại vi và bộ nhớ được thay bằng một đơn vị điều khiển điện tử đặc biệt là chip truy nhập bộ nhớ trực tiếp DMAC (direct memory access controller). Phương pháp này cho phép vào/ra số liệu rất nhanh. Tốc độ chuyển số liệu chỉ bị hạn chế bởi thời gian truy nhập của bộ nhớ. CPU không cần đọc, giải mã và thực hiện các lệnh chuyển số liệu, các lệnh điều chỉnh địa chỉ và các lệnh kiểm tra độ dài mảng đã chuyển để biết thời điểm kết thúc.

Nó hoàn toàn chuyển quyền điều khiển bus cho chip DMAC và số liệu được truyền trực tiếp giữa ngoại vi và bộ nhớ dưới sự điều khiển của chip này. Chip DMAC sẽ tạo địa chỉ, các tín hiệu điều khiển đọc/viết ngoại vi, tính số từ đã chuyển và thông báo khi chuyển xong mảng số liệu.

Hình 4-9 cho thấy có các đường liên hệ trực tiếp giữa bộ nhớ chính và các ngoại vi qua chip DMAC mà không qua CPU.

2.2. Thủ tục DMA

Có hai phương pháp DMA:



Hình 4-9

- Phương pháp thông dụng hơn là bắt CPU phải tự treo (trở về trạng thái trở kháng cao) và nhường quyền điều khiển bus cho chip DMAC. Thay vì gửi yêu cầu ngắt đến CPU, ngoại vi gửi đến chip DMAC yêu cầu về số liệu DRQ. Lúc đó chip DMAC phát ra tín hiệu đặc biệt HOLD (HRQ) gửi tới CPU. Thông báo cho CPU về yêu cầu thực hiện DMA. CPU sẽ kết thúc chu kỳ máy hiện tại, thông báo cho chip DMAC trạng thái của mình bởi tín hiệu "ghi nhận treo" HLDA và sau đó tự treo, thả nối bus. Chip DMAC lấy quyền điều khiển bus, tạo địa chỉ, ghi nhận yêu cầu số liệu của ngoại vi bằng tín hiệu DACK và tạo tín hiệu điều khiển đọc/viết bộ nhớ MEMRD, MEMRW và đọc/viết ngoại vi IORD, IOWR. Số liệu lúc này được chuyển trực tiếp giữa ngoại vi và bộ nhớ. Thông thường chip DMAC chuyển liên tục cả mảng số liệu, nên trong nó có thanh đếm số từ (word count register) cho mỗi kênh. Sau mỗi lần chuyển được một từ, nội dung thanh đếm giảm đi 1, địa chỉ DMAC được hiệu chỉnh lại (hoặc tăng hoặc giảm 1). Khi nội dung thanh đếm số từ bằng 0, tại chân EOP sẽ phát tín hiệu TC (terminal count) quá trình truy nhập trực tiếp kết thúc (/EOP=0), DMAC ngừng phát tín hiệu HOLD (HOLD=0) trả vi xử lý về trạng thái hoạt động bình thường, vi xử lý nắm quyền điều khiển bus và ngừng phát tín hiệu HLDA trả 8237 về trạng thái không hoạt động (SI)

- Lấy lén chu kỳ (cycle stealing): Chip DMAC sử dụng những chu kỳ mà CPU không truy nhập bộ nhớ, như vậy nó có thể sử dụng bus mà không cần thông báo cho CPU. Nhược điểm của phương pháp này là yêu cầu về thời gian rất khắt khe.

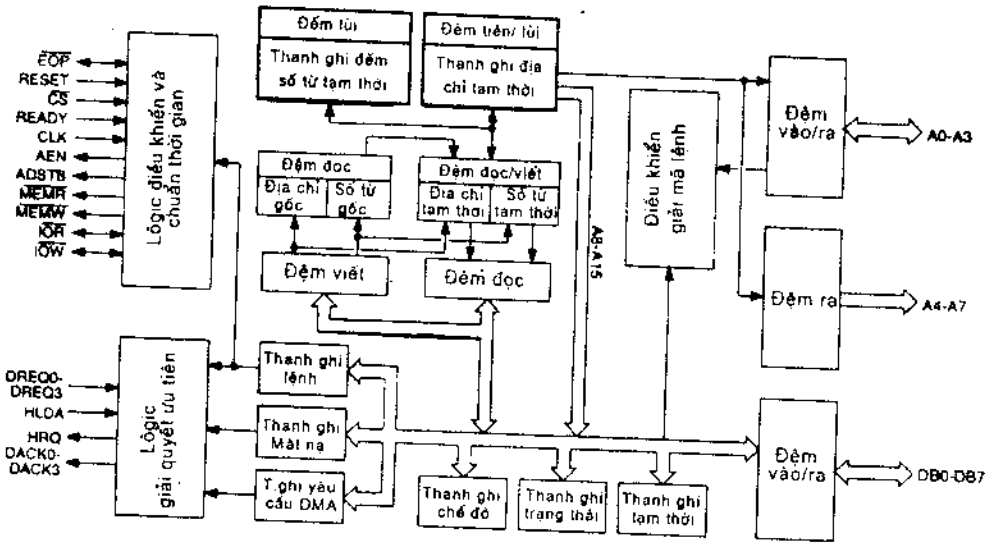
2.3. Sơ đồ khối của chip DMA-8237

Hình 4-10 là sơ đồ bên trong chip DMA-8237 gồm ba khối chính:

Khối điều khiển và phân chia thời gian cho các hoạt động bên trong và tạo tín hiệu điều khiển cho bên ngoài.

Khối điều khiển giải mã lệnh do CPU đưa tới trước khi phục vụ DMA và giải mã từ điều khiển chế độ để chọn kiểu DMA.

Khối mã hóa ưu tiên làm trọng tài giải quyết ưu tiên cho những kênh DMA yêu cầu phục vụ trong cùng một thời điểm.



Hình 4-10

Hình 4-11 là sơ đồ chân các tín hiệu:

IOR (chân số 1): Chỉ thị rằng chip DMAC đang đọc số liệu từ ngoại vi qua địa chỉ cảng.

IOW (chân số 2): DMAC đang viết số liệu tới ngoại vi qua cảng.

MEMR (chân số 3): Số liệu được đọc từ bộ nhớ chính.

MEMW (chân số 4): Số liệu đang được viết vào bộ nhớ chính.

READY (chân số 6): Tín hiệu vào từ bộ nhớ hoặc ngoại vi.

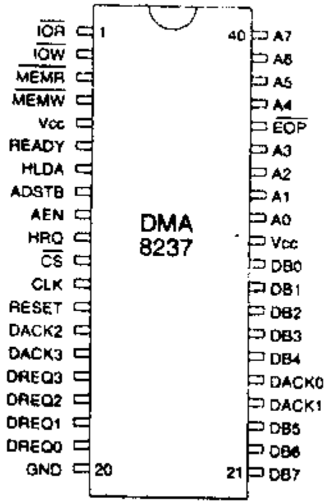
HLDA (chân số 7): CPU hoặc bus master báo cho biết nó đã rời khỏi bus cho DMAC.

ADSTB (chân số 8): Nhận byte cao vào chốt địa chỉ ngoài của DMAC.

AEN (chân số 9): Kích hoạt chốt địa chỉ của DMAC.

HRQ (chân số 10): Tín hiệu yêu cầu DMA từ chip DMA-8237.

$\overline{CS}$ (chân số 11): Tín hiệu chọn chip.



Hình 4-11

CLK (chân số 12): Lối vào xung nhịp đồng hồ (4,77 MHz hoặc 7,16MHz trong PC)

RESET (chân số 13)

DACK0-DACK3 (chân số 25, 24, 14, 15): Chấp nhận DMA

DREQo - DREQ3 (chân số 19-16): Đòi hỏi DMA từ ngoại vi

DB0 -DB7 (chân số 30-26, 23-21): Bus số liệu 2 hướng.

A_0-A_3 (chân số 32-35): Ở trạng thái standby, CPU dùng nibble này để định địa chỉ các thanh ghi. Khi 8237 được kích hoạt, đây là 4 bit địa chỉ thấp.

A_4-A_7 (chân số 37-40): Ở trạng thái standby, 4 chân này bị cấm. Khi DMA-8237 được kích hoạt, đây là 4 bit địa chỉ.

EOP (chân số 36): Là tín hiệu hai chiều. Khi là đầu vào nó được dùng để buộc DMAC kết thúc xử lý DMA. Khi là đầu ra nó phát tín hiệu kết thúc truyền số byte TC (terminal count) mà một kênh DMA đảm nhiệm.

Vcc (chân số 5, 31): Nguồn nuôi +5V

GND (chân số 20): Nối đất OV

Các thanh ghi được chia thành 2 nhóm:

- Nhóm các thanh ghi điều khiển, trạng thái

- Nhóm các thanh ghi làm việc.

Bảng sau liệt kê các thanh ghi đó.

Tên thanh ghi	Độ dài (bit)	Số lượng
Địa chỉ gốc	16	1
Số từ gốc	16	1
Địa chỉ tức thời	16	1
Số từ tức thời	16	1
Trạng thái	8	1
Lệnh	8	1
Tạm thời	8	1
Chế độ	6	1
Mặt nạ	4	1
Yêu cầu DMA	4	1

Máy PC/XT chỉ dùng một chip DMAC. Trong máy PC/AT (16 bit) sử dụng 2 chip DMAC ghép nối theo chế độ chủ tớ để tăng kênh truy nhập. DMAC chủ coi DMAC tớ như là một thiết bị ngoại vi. Các chân HRQ và HLDA của chip DMAC tớ được nối với kênh 0 của chip chủ để nhằm cho các kênh 0-3 có mức ưu tiên cao hơn 3 kênh của chip chủ. Các kênh 0-3 của chủ được quy ước là kênh DMA từ 4-7 của máy AT. Bảng sau liệt kê ứng dụng của những kênh DMA trong máy vi tính cá nhân.

Kênh	Bộ điều khiển	Độ rộng	Dùng cho
0	1	8 bit	Làm tươi bộ nhớ
1	1	8 bit	Bản mạch âm thanh hay mạng...
2	1	8 bit	Điều khiển đĩa mềm
3	1	8 bit	PC/XT : bộ điều khiển đĩa cứng AT : dự trữ
4	2	16 bit	nối DMAC tớ sang DMAC chủ
5	2	16 bit	Dự trữ
6	2	16 bit	Dự trữ
7	2	16 bit	Dự trữ

2.4. Truy nhập chip DMA8237

8237 làm việc trong hai chu kỳ máy cơ bản: Chu kỳ nghỉ và chu kỳ hoạt động. Chu kỳ nghỉ xảy ra khi không có yêu cầu DMA, 8237 ở chu kỳ nghỉ bằng cách liên tục thực hiện trạng thái SI. Nó kiểm tra các đường yêu cầu của số liệu DRQ vào tất cả các chu kỳ đồng hồ, đồng thời kiểm tra chân chọn mạch CS để xem CPU có ý định gửi lệnh đến hoặc đọc/ghi thông số, trạng thái, chế độ DMA từ (hoặc vào) chip DMAC hay không. Nếu CS = 0 và HRQ =0, 8237 tuân theo sự điều khiển bằng chương trình của CPU. A_0-A_3 là những đầu vào địa chỉ để chọn các thanh ghi trong của 8237. Những thanh ghi này được ghi vào hoặc đọc ra tuỳ thuộc vào tín hiệu điều khiển đọc/viết /IORD và /IOWD của CPU.

Chu kỳ hoạt động bắt đầu lúc chip DMAC phát yêu cầu treo HRQ cho CPU. Số liệu được chuyển bằng DMA trong những chu kỳ này theo một trong những chế độ sau:

Chế độ chuyển từng từ (single transfer mode)

Chế độ chuyển mảng (block transfer)

Chế độ chuyển theo nhu cầu (demand transfer)

Chế độ ghép tầng (cascade mode)

Mỗi chu kỳ máy DMA bao gồm một số trạng thái. Có tất cả 7 trạng thái, mỗi trạng thái chiếm một chu kỳ nhịp đồng hồ.

SI: Trạng thái không hoạt động. 8237 ở trạng thái này khi không có yêu cầu DMA hợp lệ, chip chỉ nhập lệnh từ CPU khi đang ở trạng thái này.

S_0: Là trạng thái đầu của quá trình phục vụ DMA, khi chip DMAC đã gửi yêu cầu treo cho CPU nhưng chưa nhận được tín hiệu ghi nhận treo.

S_1, S_2, S_3, S_4: Các trạng thái làm việc của DMAC. Lúc này chip DMAC trực tiếp điều khiển chuyển số liệu bằng các địa chỉ, tín hiệu điều khiển đọc/viết do chính mình tạo ra.

SW: Là trạng thái đợi được xen vào giữa S_3 và S_4 do tác động của đầu vào READY của chip DMAC. SW cần thiết khi thời gian thâm nhập của bộ nhớ quá lớn.

Trước khi để DMAC làm việc, CPU cần phải lập cho nó trật tự ưu tiên, mặt nạ cho từng kênh... và quy định cho từng kênh địa chỉ DMA, độ dài mảng (số đếm từ) bằng cách ghi vào các thanh ghi trong của 8237 tuân theo quy định

về mã của chúng. Sau đây mô tả các thanh ghi lệnh, thanh ghi chế độ, thanh ghi yêu cầu và thanh ghi mặt nạ. Tất cả các thanh ghi này đều có dung lượng 8 bit.

Thanh ghi yêu cầu: Hai bit thấp D_0, D_1 dùng để chọn kênh. Giá trị thập phân của hai bit này là số kênh. Bit D_2(1/0) = lập/xoá bit yêu cầu. Các bit còn lại không sử dụng.

Thanh ghi lệnh: D_0(1/0) = cho phép/cấm từ bộ nhớ đến bộ nhớ.
D_1(1/0) = cho phép/cấm địa chỉ kênh 0.
D_2(1/0) = cho phép/cấm DMA.
D_4(1/0) = ưu tiên vòng/cố định.
D_6(1/0) = mức tích cực của DRQ là cao/thấp.
D_7(1/0) = mức tích cực của DACK là cao/thấp.

Thanh ghi chế độ : D_0, D_1 dùng để chọn kênh như thanh ghi yêu cầu.
D_2, D_3(00-11) = chế độ kiểm tra/đọc /viết.
D_4(1/0) = cho phép/c m tự khởi động.
D5(1/0) = đếm tăng/giảm địa chỉ.
D_6, D_7(00-11) = chuyển theo nhu cầu/từng từ/mảng/tầng.

Thanh ghi mặt nạ: D_0(1/0) = lập/xoá mặt nạ kênh 0, tương tự D_1, D_2, D_3 lần lượt cho kênh 1, 2, 3. Các bit còn lại không dùng.

Trong quá trình hoạt động, 8237 luôn cập nhập trạng thái của mình vào thanh ghi trạng thái. Thông tin trạng thái cho biết kênh DMA nào đã đạt đến số đếm kết thúc TC (terminal count) tức là đã chuyển xong mảng số liệu có độ dài quy định ở thanh ghi số đếm từ gốc, hoặc bị bắt buộc kết thúc chuyển do tác động của tín hiệu EOP đang chờ phục vụ ở bất cứ kênh nào nếu có. Ngoài ra còn có 2 lệnh đặc biệt:

- Lệnh xoá thanh ghi flip-flop trong nhằm để đưa về trạng thái ban đầu trước khi đọc hoặc viết địa chỉ hoặc số đếm từ mới vào 8237 (xem ví dụ cuối chương), bằng cách đưa một byte bất kỳ ra thanh ghi này.

- Xoá toàn bộ (master clear) có chức năng như RESET lệnh: tất cả các thanh ghi lệnh, trạng thái, yêu cầu, mạch lật trong đều bị xoá, thanh ghi mặt nạ được lập và 8237 chuyển sang chu kỳ nghỉ.

Bảng dưới đây liệt kê các lệnh của chip 8237

Tín hiệu						Lệnh
A_3	A_2	A_1	A_0	/IOR	/IOW	
1	0	0	0	0	1	Đọc thanh ghi trạng thái
1	0	0	0	1	1	Nạp thanh ghi lệnh
1	0	0	1	0	1	Không hợp lệ
1	0	0	1	1	0	Nạp thanh ghi yêu cầu DMA
1	0	1	0	0	1	Không hợp lệ
1	0	1	0	1	0	Ghi từng bit cho thanh ghi mặt nạ
1	0	1	1	0	1	Không hợp lệ
1	0	1	1	1	0	Nạp thanh ghi chế độ
1	1	0	0	0	1	Không hợp lệ
1	1	0	0	1	0	Xoá mạch lật trong
1	1	0	1	0	1	Đọc thanh ghi tạm thời
1	1	0	1	1	0	RESET DMA
1	1	1	0	0	1	Không hợp lệ
1	1	1	0	1	1	Không hợp lệ
1	1	1	1	0	1	Không hợp lệ
1	1	1	1	1	1	Ghi toàn bộ thanh ghi mặt nạ

Bảng sau liệt kê địa chỉ vào/ra thanh ghi điều khiển và trạng thái của DMA

DMA(1)	DMA(2)	Đọc/viết (R/W)	Thanh ghi
08h	D0h	R	Trạng thái
08h	D0h	W	Lệnh
09h	D2h	W	Yêu cầu DMA
0Ah	D4h	W	Che kênh
0Bh	D6h	R	Chế độ DMA
0Fh	DEh	W	Mặt nạ

(1): chủ trong PC/XT, tớ trong AT; (2): chủ trong AT

Các địa chỉ vào/ra của thanh ghi địa chỉ và đếm số từ được liệt kê dưới đây:

DMA (1)	DMA (2)	Thanh ghi
00h	C0h	địa chỉ kênh 0/4
01h	C1h	đếm kênh 0/4
02h	C2h	địa chỉ kênh 1/5
03h	C3h	đếm kênh 1/5
04h	C4h	địa chỉ kênh 2/6
05h	C5h	đếm kênh 2/6
06h	C6h	địa chỉ kênh 3/7
07h	C7h	đếm kênh 3/7

(1) : chủ trong PC/XT, tớ trong AT, (2) : chủ trong AT

Ví dụ sau đưa giá trị 1080h vào thanh ghi địa chỉ kênh 6 và giá trị 0100h vào thanh ghi đếm tương ứng của DMA chủ:

out d8h, al; đưa một giá trị bất kỳ ra flip-flop để đưa nó về trạng thái ban đầu.
out c4h, 80h; đưa byte thấp của giá trị địa chỉ ra thanh ghi địa chỉ.
out c4h, 10h; đưa byte cao.
out d8h, al; đưa một giá trị bất kỳ ra flip-flop để đưa nó về trạng thái ban đầu.
out c5h, 00h; đưa byte thấp của giá trị địa chỉ ra thanh ghi đếm.
out c5h, 01h; đưa byte cao.

8237 có thể làm việc ở hai chế độ ưu tiên khác nhau theo quy định phần mềm từ CPU (nạp vào thanh ghi lệnh). Chế độ ưu tiên cố định: kênh 0 có mức ưu tiên cao nhất, kênh 3 có mức thấp nhất. Chế độ ưu tiên vòng: kênh vừa được phục vụ sẽ có mức ưu tiên thấp nhất, lúc mới lập chế độ kênh 0 ở mức cao nhất.

Bài tập chương 4

1. Trình bày cách tính số lượng đoạn nhớ logic trong bộ nhớ qui ước (1MB đầu tiên).
2. Đổi địa chỉ nhớ logic sau đây ra địa chỉ thực : AC1B : D23C; F16E : 5D3C
3. Lập bảng tính địa chỉ véc tơ ngắt của 256 ngắt mềm (INT 00h - INT FFh)

Chương 5

BỘ NHỚ NGOÀI

Mục tiêu : Bộ nhớ chính bằng vật liệu bán dẫn trên bản mạch chính không thể lưu trữ một khối lượng rất lớn các thông tin, do vậy cần phải có thêm các thiết bị nhớ bên ngoài như bằng giấy đục lỗ, băng cassette, trống từ, đĩa từ, đĩa quang. Những thiết bị lưu trữ dữ liệu như vậy được gọi là bộ nhớ ngoài. Thiết bị nhớ ngoài thông dụng nhất là đĩa từ và ổ đĩa từ. Đĩa từ là một tấm đĩa tròn mỏng làm bằng chất dẻo mylar, hay bằng thuỷ tinh hoặc bằng kim loại cứng, trên có phủ một lớp bột từ tính ô-xít sắt từ. Đĩa từ sử dụng kỹ thuật ghi từ để lưu trữ dữ liệu, đó là việc định hướng các domain từ để tạo ra các bit thông tin 0 và 1. Khi đã được ghi lên đĩa, dữ liệu có thể tồn tại cả khi cắt điện PC. Tuy nhiên, giống như băng từ, dữ liệu cũ cũng có thể được xoá đi, thay thế bởi dữ liệu mới nhiều lần. Trong chương này ngoài đĩa từ, sẽ giải thích cho sinh viên hiểu được đĩa quang và các ổ đĩa quang, đặc biệt là cách thức hệ điều hành quản lý các tập tin (file) trên đĩa từ.

I. ĐĨA TỪ

1. Tổ chức vật lý và cấu tạo của đĩa từ

Đĩa mềm (floppy disk) thường gồm một đĩa từ bằng nhựa dẻo (có thể uốn cong) được bảo vệ bởi một bao giấy hoặc nhựa cứng. Trên bao có khoét một lỗ dài cho phép để đầu đọc/viết (head) của ổ đĩa có thể tiếp xúc với mặt đĩa để đọc hoặc viết dữ liệu. Với loại đĩa có đường kính 5,25 inch, dung lượng chứa tin là 360 kB hoặc 1,2MB. Với loại đĩa có đường kính 3,5 inch, dung lượng chứa tin là 720kB hoặc 1,44MB.

Mỗi đĩa từ nói chung được tổ chức thành các đơn vị như sau:

-Track (rãnh từ): Là các vùng đường tròn đồng tâm mà dữ liệu được ghi lên đó. Mật độ ghi tin được tính theo đơn vị track/inch.

- Sector (cung từ): Mỗi track được chia nhỏ thành nhiều sector, mỗi sector chứa được 512 byte dữ liệu. Số sector trên một track tuỳ thuộc loại đĩa. Đĩa mềm 3,5 inch có 18 cung trên một rãnh. Do đó sector trên các rãnh khác nhau có thể không dài bằng nhau. Tuy nhiên, do đĩa cứng có rất nhiều rãnh và các chu vi rãnh phía ngoài lớn hơn chu vi các rãnh phía trong, nếu chia mỗi rãnh theo cùng một số cung nhất định thì mật độ thông tin ở phía ngoài nhỏ gây lãng phí. Để tránh điều đó, mặt đĩa được chia thành nhiều vùng (zone) gồm một số rãnh và các rãnh trong một vùng có cùng một số cung.

- Cluster (liên cung): Là một nhóm gồm một hay nhiều sector, tùy theo quy định của hệ điều hành.

Đĩa từ có thể được đọc từ hai mặt, nên có hai đầu từ ở hai phía. Tất cả các track, sector được đánh số theo thứ tự tăng dần như sau:

- Track được đánh số khởi đầu từ 0 kể từ vòng ngoài vào

- Đầu từ (head) cũng được đánh số từ 0 đi từ trên xuống dưới.

Trong khi đó sector được đánh số từ 1 trở đi.

Bảng sau mô tả các dạng trữ tin của đĩa mềm loại 5,25 và 3,5 inch

Loại đĩa mềm	Khả năng trữ tin	Số track	Số sector/track	Tổng số sector	Track/inch
5,25" SS/SD	160kB	40	08	320	48
5,25" SS/SD	180kB	40	09	360	48
5,25" DS/DD	320kB	40	08	640	48
5,25" DS/DD	360kB	40	09	720	48
5,25" DS/HD	1,2MB	80	15	2400	48
3,50" DS/DD	720kB	80	09	1440	135
3,50" DS/HD	1,44MB	80	18	2880	270
3,50" DS/ED	2,88MB	80	36	5760	540

SS = Single Side (một mặt); HD = High Density (mật độ cao);
DS = Double side (hai mặt); ED = Extra High Density (mật độ cực cao);
SD = Single Density (mật độ đơn); QD = Quardruple Density (mật độ gấp bốn);
DD = Double Density (mật độ kép).

Ổ đĩa từ có hai động cơ, một để quay đĩa và một động cơ bước để dịch đầu từ theo các rãnh đến rãnh từ cần đọc.

Đĩa cứng (hard disk, fixed disk) thường gồm một hoặc nhiều đĩa từ bằng kim loại hoặc nhựa cứng được sắp thành một chồng theo một trục đứng và được đặt trong một hộp kín để tránh bụi. So với đĩa mềm, dung lượng nhớ của nó lớn hơn nhiều, thường từ 10 Mbyte đến vài chục Gbyte. Ổ đĩa cứng có nhiều đầu từ, các đầu từ này cùng được gắn trên một cần truy xuất và được di chuyển thành một khối. Khi đĩa quay vòng, đầu từ không đụng lên mặt đĩa mà bay là là trên mặt đĩa cách nhau bởi một lớp đệm không khí. Đệm này sinh ra khi đĩa chuyển động nhanh. Khi phanh ổ đĩa, đĩa quay chậm lại, hiệu ứng đệm không khí giảm nên đầu từ rất dễ va chạm vào mặt đĩa. Để tránh điều này đầu từ được đưa về một vị trí an toàn phía trong hay ngoài vùng dữ liệu (land zone). Khoảng cách giữa mặt đĩa và đầu từ tuỳ thuộc tốc độ quay cũng như mật độ ghi tin của đĩa. Khoảng cách này rất nhỏ đối với đĩa cứng, khoảng 0,0003 mm. Để so sánh, vân tay và hạt bụi thuốc lá có kích thước 0,003 mm, còn đường kính sợi tóc cỡ 0,01 mm. Do vậy phải đặt các đĩa từ này trong một hộp kín để tránh bụi.

Đĩa cứng cũng được phân thành các đơn vị vật lý như đĩa mềm, nhưng ở đây có thêm một khái niệm nữa là cylinder (từ trụ). Vì chồng đĩa cứng có nhiều mặt, nên vị trí của các đầu từ khi di chuyển trên các mặt đó sẽ tạo thành một mặt trụ (cylinder), đó là một chồng các track sắp nằm lên nhau đối với một vị trí đầu từ.

Tốc độ quay của đĩa cứng cũng như tốc độ di chuyển đầu từ nhanh hơn nhiều so với đĩa mềm. Tốc độ của đĩa mềm khoảng 300 -360 vòng/phút tuỳ loại đĩa, còn của đĩa cứng hiện nay cỡ 3600 vòng/phút đến 8000 vòng/phút. Thời gian truy xuất dữ liệu (data access time) của ổ đĩa là một thông số quan trọng ảnh hưởng mạnh mẽ đến việc xử lý số liệu trong PC, thời gian này là tổng hợp của thời gian tìm kiếm (seek time), thời gian chuyển đầu từ (head switch time) và thời gian quay trễ (rotaional latency). Thời gian tìm kiếm là thời gian chuyển dần từ một track này sang track khác; thời gian tìm kiếm chậm nhất ứng với hai track trong và ngoài cùng của đĩa; thời gian tìm kiếm trung bình là bằng thời

gian cần có để đặt đầu từ tới một vị trí được yêu cầu ngẫu nhiên. Thời gian chuyển đầu từ là thời gian ổ đĩa phải bỏ ra để chuyển giữa hai trong số các đầu từ khi đọc hay viết dữ liệu. Thời gian quay trễ là thời gian tính từ khi đầu từ được đặt lên một track đến khi nó tới được sector mong muốn. Tất cả các thông số trên đều được đo bằng mili giây (ms). Ngày nay có thể phân loại như sau:

- Thời gian truy xuất từ 28ms đến 40ms là trung bình,
- Trên 40 ms là loại chậm,
- Từ 18 ms đến 28 ms là loại nhanh,
- Dưới 18 ms là loại cực nhanh.

2. Phương pháp mã hoá và định dạng

Để tránh lỗi trong quá trình ghi tin lên đĩa từ do nhiều nguyên nhân, đặc biệt như sai số của tốc độ chuyển động giữa đầu từ và đĩa, thông tin cần phải được đồng bộ mã hóa, giải mã khi đọc và ghi. Đĩa từ thường dùng các phương pháp mã hoá cho mật độ ghi tin cao như FM (frequency modulation), MFM (modified frequency modulation), GCR (group coded recording), RLL (run length limited).

Phương pháp FM sử dụng xung đồng hồ tần số f (250KHz) để làm tín hiệu đồng bộ, khoảng cách giữa hai xung gọi là một ô bit, xung số liệu được ghi ở giữa ô bit. Nếu có xung số liệu thì bit số liệu bằng 1, còn nếu không có xung số liệu thì bit số liệu bằng 0. Như vậy, tuỳ theo sự phân bố của số liệu theo thời gian mà tần số của dãy xung tổng hợp của xung đồng hồ và xung số liệu sẽ thay đổi và đạt cực đại là 2f khi tất cả các số liệu bằng 1. Nhược điểm của phương pháp FM là độ dư thừa thông tin tới 50%, do đó làm chậm tốc độ truyền dữ liệu.

Khắc phục nhược điểm đó, phương pháp MFM đề xuất: Chỉ ghi xung đồng hồ ở đầu mỗi ô bit (thời gian xung đồng hồ nhỏ) nếu trong ô bit này và ô bit trước đó bit số liệu cùng bằng 0. Như vậy xung đồng hồ sẽ xuất hiện liên tục khi các bit số liệu liên tục bằng 0. Bằng cách đó, với khoảng cách tối đa 1,5 ô bit cực từ đã được đổi, đảm bảo tần số đổi cực đủ lớn để tín hiệu đọc có khả năng tái định thời (tái đồng bộ). Tiếp theo là phương pháp gần giống như thế M^2FM : quy định xung đồng hồ chỉ xuất hiện tại ô bit nếu tại ô bit đó và các ô trước đó và sau đó bit số liệu đều bằng 0, do vậy tăng được tốc độ truyền số liệu hơn.

Để tăng tốc độ truyền cao hơn nữa và tiết kiệm không gian đĩa, phải loại bỏ hẳn xung đồng bộ, nhưng phải duy trì việc thay đổi cực từ hay mức tín hiệu với một tần số đủ lớn để tín hiệu đọc căn cứ vào đó mà tái đồng bộ. Phương

pháp GCR đổi một nhóm mã số liệu 4b thành một nhóm 5b (xem bảng sau) và thay vì ghi nhóm mã số liệu lên đĩa từ lại ghi nhóm mã 5b này. Khi đọc sẽ giải mã trở lại.

4 bit dữ liệu	5 bit mã	4 bit dữ liệu	5 bit mã
0000	11001	1000	11010
0001	11011	1001	01001
0010	10010	1010	01010
0011	10011	1011	01011
0100	11101	1100	11110
0101	10101	1101	01101
0110	10110	1110	01110
0111	10111	1111	0111

Như vậy, trong nhóm 5 bit mã không có quá hai bit liên tục bằng 0, mỗi nhóm đều chứa ít nhất một số 0 hay một số 1. Mã có khoảng cách đổi cực từ nhỏ nhất là 1 và lớn nhất là 8 (khi mã ở hàng 16 có trước hàng13). Khoảng cách tối đa tương đối ngắn (8) nên mã này có khả năng tự định thời. Phương pháp này được đánh giá làm tăng mật độ ghi tin gấp 2 lần phương pháp FM. Mã GCR chủ yếu được dùng trong casset từ. Còn đĩa mềm dùng mã MFM.

Mã RLL được định nghĩa là một nhóm mã với khoảng cách nhỏ nhất và lớn nhất nhất định giữa mỗi lần đổi cực từ. Khoảng cách lớn nhất phải đảm bảo sao cho tần số đổi cực đủ lớn để tín hiệu đọc có khả năng tái định thời. Theo định nghĩa này thì mã GCR được coi là mã RLL1,8. Số 1,8 nghĩa là khoảng cách nhỏ nhất và lớn nhất bằng 1và 8. Có thể nói về bản chất thì tất cả các mã đề cập trên đều là mã RLL. Tuy nhiên mã dùng cho đĩa cứng là mã RLL2,7. Mã này chia chuỗi bit dữ liệu thành các nhóm 2, 3, 4 bit và mỗi nhóm đó được đổi thành nhóm mã có số ký tự gấp 2 bao gồm các ký tự S (space) và R (reversal). Mã tín

hiệu khi từ cực không đổi gọi là S và mã khi từ đổi cực là R. Bảng sau liệt kê 7 nhóm bít số liệu và các nhóm mã S/R tương ứng.

Nhóm bit dữ liệu	Nhóm mã S/R
10	SRSS
11	RSSS
000	SSSRSS
010	RSSRSS
011	SSRSSS
0010	SSRSSRSS
0011	SSSSRSSS

Mã RLL2,7 luôn đảm bảo có ít nhất hai mã S ở giữa hai mã R và các nhóm mã luôn kết thúc với ít nhất 2 mã S, do vậy khoảng cách tối thiểu là 2. Khoảng cách tối đa là 7 vì không có nhóm mã nào bắt đầu với nhiều hơn 4 mã S và kết thúc với lớn hơn 3 mã S. Như vậy mã có khả năng tự định thời.

Mã RLL2,7 làm tăng gấp đôi chuỗi thông tin, nhưng một mã ký tự chỉ sử dụng 1/3 ô bit, do đó một ô bit có khả năng lưu trữ 1,5 bit so với 1 bit của các phương pháp khác và do vậy làm tăng dung lượng trữ tin gấp rưỡi và tốc độ truyền tin cũng cao hơn. Có thể giải thích điều này như sau: bit R ứng với sườn tăng hoặc giảm của tín hiệu (quá trình quá độ khi đổi cực từ), còn bit S ứng với mức tín hiệu dù là mức cao hay thấp (cực từ cố định), một ô bit chứa hai mã S do không có xung đồng hồ như ở các phương pháp khác, giữa hai mã S có thể chứa thêm được một mã R - sườn xung nữa. Như vậy một ô bit có thể có 3 bit, có thể kiểm tra điều này với tất cả các bộ mã S/R nói trên, lưu ý rằng có những trường hợp phải trải các mã trên vài ô bit và chia trung bình mới thấy kết quả trên.

Muốn đĩa sử dụng được bởi hệ điều hành, ví dụ với DOS, 3 bước sau phải

được thực hiện:

- Định dạng cấp thấp (low-level format): Phân định ra các track và các sector cũng như ghi các thông tin nhận dạng và đồng bộ trên chúng.
- Chia đĩa thành các phân khu (partitions), bước này chỉ thực hiện với đĩa cứng với lệnh FDISK.EXE.
- Định dạng cấp cao (high-level-format): Bằng lệnh FORMAT.COM, DOS tạo ra cấu trúc dữ liệu logic của các phân khu mà nó sẽ sử dụng về sau để ghi giữ vị trí các tệp tin được đặt trên đĩa. Quá trình này sẽ tạo ra sector khởi động, hai bảng FAT và thư mục gốc.

Trong phần này sẽ nghiên cứu bước định dạng cấp thấp, các bước sau là các bước có tính logic sẽ đề cập đến trong các phần ngay sau đây.

Định dạng cấp thấp (low-level format) nhằm tạo ra những track và sestor trên đĩa bằng cách viết lên đĩa những thông tin liên quan đến chúng. Nói chung các thông tin này là giống nhau cho cả đĩa mềm và đĩa cứng, chỉ có một ít khác biệt sẽ được lưu ý. Với các sector, những thông tin này được ghi vào một vùng gọi là tiêu đề nhận dạng sector. Nó được đặt nằm ở đầu mỗi sector. Vùng tiêu đề này chứa các thông tin như sau: số thứ tự đầu từ, số sector, số cylinder, dấu khai báo nhận dạng ID bắt đầu từ đâu và một mã ECC dùng để phát hiện lỗi dữ liệu. Mạch điều khiển ổ đĩa sẽ sử dụng những thông tin trên tiêu đề để tìm đến đúng sector mà nó nhận được lệnh phải tìm.

Một rãnh đĩa cứng chứa các thông tin điển hình như thí dụ sau:

Định dạng kiểu MFM	Định dạng kiểu RLL
Phần mở đầu rãnh gồm :	Phần mở đầu rãnh gồm :
10 byte để đồng bộ SYNC: chứa mã 00h, có nhiệm vụ đồng bộ hoá bộ điều khiển với tốc độ quay của đĩa.	11 byte để đồng bộ SYNC: chứa mã 00h
2 byte đánh dấu địa chỉ chỉ số IAM (index address mark) chứa mã: A1F, FCh, báo cho bộ điều khiển biết một cung sắp bắt đầu.	2 byte đánh dấu IAM : A1h, FCh
11 byte khe phân cách đầu tiên GAP1: chứa mã 4Eh. báo và dành thời gian cho bộ điều khiển tìm ra đầu cung.	12 byte khe phân cách đầu tiên GAP1 : FFh

Tiếp theo là cung thuộc rãnh hiện hành, bắt đầu và gồm các tin sau:	Bắt đầu cung:
7 byte SPD : 4Eh	
10 byte đồng bộ SYNC chứa mã 00h để tái đồng bộ	10 byte đồng bộ SYNC : 00h
2 byte dấu địa chỉ nhận dạng IDAM (identification address mark) chứa mã A1h, FEh báo thông tin tiếp theo chứa địa chỉ của cung	2 byte dấu IDAM: 5Eh, A1h
4 byte nhận dạng ID : số trụ, đầu từ, cung và cờ	4 byte nhận dạng ID : số trụ, đầu từ, cung và cờ
4 byte mã ECC của 4 byte ID	4 byte mã ECC của 4 byte ID
5 byte khe phân cách thứ hai GAP2 chứa mã 00h	5 byte khe phân cách thứ hai GAP2 chứa mã 00h
10 byte đồng bộ SYNC xuất hiện lại để chỉnh lại tốc độ đọc và chứa mã 00h	11 byte đồng bộ SYNC : 00h
2 byte DAM (data address mark) chứa mã A1h, F8h báo tin vùng tới sẽ là vùng dữ liệu	2 byte đánh dấu DAM : 5Eh, A1h
512 byte lưu trữ dữ liệu (data)	512 byte lưu trữ dữ liệu (data)
4 byte mã ECC cho dữ liệu	4 byte ECC cho dữ liệu
15 byte khe phân cách 3 (GAP3) chứa mã 00h có nhiệm vụ đặc biệt, đây là vùng đệm giữa hai cung đảm bảo	3 byte 00h và 17 byte FFh làm vùng đệm phân cách 3 (GAP3)
Kết thúc rãnh (EOT)	Kết thúc rãnh (EOT)
56 byte 00h vùng phân cách GAP4	93 byte 00h vùng phân cách GAP4

Các thông tin trên một rãnh của đĩa mềm chỉ có một vài điểm khác so với của đĩa cứng như số byte dành cho mỗi đoạn tin là khác nhau, rãnh đĩa mềm bắt đầu bằng khoảng trống GAP4A khá dài với dung lượng 80 byte 4Eh cho loại mã hoá MFM và 40 byte FFh cho loại mã hoá FM. GAP4A báo và dành thời gian cho bộ điều khiển tìm ra đầu rãnh. Đặc biệt, trong 4 byte ID của đĩa mềm không có cờ. Trong đĩa cứng một byte ID dùng để chỉ thị 8 cờ (1cờ=1b). Nếu bit 0 (b0) bằng 1 thì cung bị hỏng, b1=1thì toàn bộ rãnh bị hỏng, b2=1 rãnh bị hỏng và đã được thay thế bằng rãnh khác và địa chỉ trong ID chỉ đến rãnh thay thế này, b3=1 thì rãnh hiện hành là một rãnh thay thế. Trong quá trình chế tạo, có thể có rãnh hoặc cung bị hỏng, nhà sản xuất dự trữ một vài rãnh để thay thế. Danh sách các cung, rãnh có lỗi lưu trữ tại rãnh -1 và chỉ có bộ điều khiển đĩa truy nhập được, ngay cả hệ điều hành cũng không thể.

Một điểm khác nhau nữa phải nêu ra ở đây là mã phát hiện lỗi. Đĩa mềm dùng mã CRC (cycle redundancy check), nguyên lý như sau: một nhóm bit khi được lấy vào kiểm tra sẽ được gán cho các trọng số từ cao đến thấp, ví dụ 1101 gán thành $1.x^3+1.x^2+0.x^1+1.x^0$ chia cho đa thức sinh chuẩn do CCITT quy định là $x^{16}+x^{12}+x^5+1$, nghĩa là nhóm bit ứng với đa thức sinh (số chia) có các bit bằng 1 ở bit thứ 17, 13, 6 và 1, các bit còn lại bằng 0. Số dư chính là mã CRC. Khi đọc dữ liệu từ đĩa hiệu số của số bị chia và số dư sẽ chia hết cho đa thức sinh, nếu phép chia này có dư tức là có lỗi trong nhóm dữ liệu. Đĩa cứng dùng mã ECC (error correcting code) có đa thức sinh với bậc lên tới 38, các hệ số của đa thức này bằng 1 ở các bậc 38, 28, 26, 19, 17, 10, 6, 2, 0 do vậy kiểm soát được nhóm bit dài hơn, bộ điều khiển được thiết kế để không những kiểm tra lỗi mà còn chữa lỗi trong các trường hợp đơn giản.

Hệ số đan xen (interleave factor) của các sector nhằm làm khớp tốc độ quay của đĩa từ (thường là 3600 vòng/phút = 60 vòng/giây) với tốc độ mà đầu từ có thể xử lý dữ liệu khi chúng đi qua hết một sector. Ví dụ với đĩa có 17 sector/track, đầu từ sẽ phải đọc được 512 x 17 x 60 = 522240 byte số liệu trong một giây. Tốc độ này có thể nhanh hơn tốc độ xử lý của đầu từ, bộ điều khiển hay máy tính nói chung. Trong trường hợp như vậy đầu từ phải đợi đến vòng quay tới để đọc cung tiếp theo, có thể dẫn đến làm giảm tuổi thọ của đĩa do phải đọc nhiều lần và làm chậm hệ thống. Vì vậy nếu các dữ liệu được ghi lên các sector liên tiếp thì PC không thể xử lý kịp. Do đó, sector kế tiếp để ghi dữ liệu được đặt ở sector cách đó n sector. Ta nói rằng đĩa cứng này có hệ số đan xen

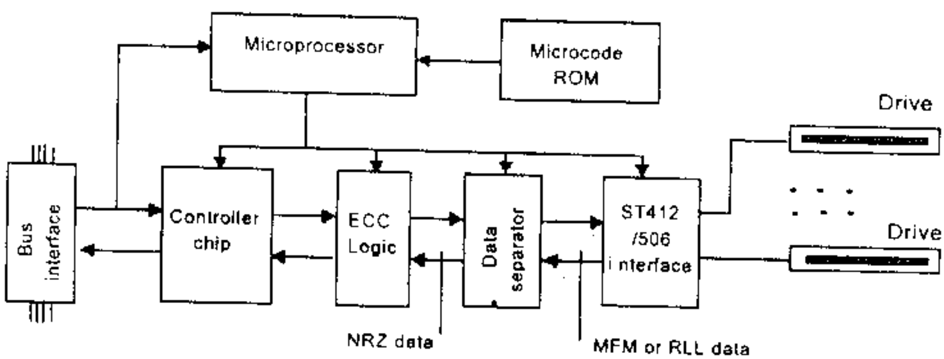
là n: 1 hay n. Nói cách khác, các sector chứa dữ liệu chỉ nối tiếp nhau về mặt logic chứ không nối tiếp nhau về mặt vật lý.

Vậy chương trình định dạng cấp thấp sẽ đánh số thứ tự các sector liên tiếp theo một trật tự định trước (phụ thuộc hệ số đan xen) không nhất thiết phải nối đuôi nhau tuần tự trên track. Với chuẩn ghép nối IDE có tốc độ truyền dữ liệu lớn nhất đến 3,3MB/s, hay 12MB/s với EIDE (enhanced IDE) thì hệ số đan xen bằng 1.

3. Bộ điều khiển và ghép nối ổ đĩa cứng. Chuẩn ghép nối

Sơ đồ khối của bộ điều khiển ổ đĩa cứng trình bày trong hình 5-1:

Các khối ở hàng đầu từ trái qua phải là vi xử lý và bộ nhớ ROM. Hàng thứ hai lần lượt là khối ghép nối bus, chip điều khiển, mạch logic kiểm tra mã ECC, bộ tách dữ liệu, bộ ghép nối ổ đĩa và cuối cùng là ổ đĩa.



Hình 5-1

Trong các chuẩn ghép nối thì chuẩn IDE (integrated drive electronics) là một giao diện được dùng rộng rãi trong các máy vi tính hiện đại và đã thay thế các chuẩn trước đó ST506 (seagate technology 506) và ESDI (enhanced small device interface).

Vào cuối 1984 hãng Compaq khởi xướng việc phát triển ghép nối IDE trên cơ sở việc gắn trực tiếp chip điều khiển ST506 vào ổ đĩa cứng. Việc nối giữa bus AT và mạch ghép nối IDE của bộ điều khiển trên các ổ đĩa cứng được thực hiện bởi bản mạch gọi là host- adapter đơn giản cắm vào một khe cắm mở rộng. Bản mạch host-adapter chỉ cung cấp một số bộ đệm và mạch giải mã dùng cho việc nối ổ đĩa IDE và bus hệ thống AT. Một cáp dẫn bẹt 40 sợi cho phép nối host-

adapter với các ổ đĩa. Độ dài cực đại của cáp là 18 inch (46cm). Sự đơn giản của bản mạch adaptor và số tín hiệu trao đổi qua cáp giảm đi làm cho chuẩn IDE hoạt động tin cậy hơn. Ghép nối IDE có thể phục vụ nhiều nhất là hai ổ đĩa (một là chủ và một là tớ, ổ đĩa chủ được gán địa chỉ 0, ổ đĩa tớ được gắn địa chỉ 1) với tốc độ truyền dữ liệu lớn nhất tới 3,3 MB/s. Dung lượng đĩa lớn nhất mà nó quản lý được là 528 MB. Chuẩn IDE nâng cao (EIDE) cho phép phục vụ 4 ổ cứng với tốc độ truyền tới 12 MB. Một số ổ đĩa IDE được trang bị thêm bộ nhớ cache cho ít nhất hai track nhằm làm giảm thời gian thâm nhập trung bình của ổ đĩa. Giao diện IDE cho phép ổ đĩa truy nhập trực tiếp bộ nhớ như là một thiết bị ngoại vi, do vậy ổ đĩa được coi là "thông minh" hơn những ổ đĩa thụ động theo các chuẩn trước đó. Dưới đây liệt kê các dây IDE và chức năng của chúng:

Tín hiệu IDE	Chân cắm	Ý nghĩa	Tín hiệu AT	Hướng
RESET	1	Reset các ổ đĩa	RESETDRV (1)	
GND	2	Nối đất (0V)	-	
DD7	3	Bus số liệu bit 7	SD7	Hai hướng
DD8	4	Bus số liệu bit 8	SD8	Hai hướng
DD6	5	Bus số liệu bit 6	SD6	Hai hướng
DD9	6	Bus số liệu bit 9	SD9	Hai hướng
DD5	7	Bus số liệu bit 5	SD5	Hai hướng
DD10	8	Bus số liệu bit 10	SD10	Hai hướng
DD4	9	Bus số liệu bit 4	SD4	Hai hướng
D11	10	Bus số liệu bit 11	SD11	Hai hướng
DD3	11	Bus số liệu bit 3	SD3	Hai hướng
DD12	12	Bus số liệu bit 12	SD12	Hai hướng
DD2	13	Bus số liệu bit 2	SD2	Hai hướng
DD13	14	Bus số liệu bit 13	SD13	Hai hướng
DD1	15	Bus số liệu bit 1	SD1	Hai hướng
DD14	16	Bus số liệu bit 14	SD14	Hai hướng
DD0	17	Bus số liệu bit 0	SD0	Hai hướng
DD15	18	Bus số liệu bit 15	SD15	Hai hướng
GND	19	Đất	-	-

(2)	20	Khoá chân cắm 20	-	-
DMARQ (3)	21	Yêu cầu DMA	DRQx	Drive → host
GND	22	Đất	-	-
DIOW	23	Viết số liệu qua kênh I/O	IOW	Host → drive
GND	24	Đất	-	-
IOR	25	Đọc số liệu qua kênh I/O	IOR	Host → drive
GND	26	Đất	-	-
IORDY (3)	27	Sẵn sàng đọc ghi	IOCHRDY	Drive → host
SPSYNC	28	Đồng bộ quay trục/ dự trữ	-	-
DMACK (3)	29	Ghi nhận DMA	DACKx	Host → drive
GND	30	Đất	-	-
INTRQ	31	Yêu cầu ngắt	IRQx	Drive → host
IOCS16	32	Truyền chế độ 16 bit qua kênh I/O	I/OCS_{16}	Drive → host
DA1	33	Bit 1 bus địa chỉ thanh ghi trong	SA1	Host → drive
PDIAG	34	Ổ đĩa sẵn sàng làm việc	Drive → drive	
DA0	35	Bit 0 bus địa chỉ thanh ghi trong	SA0	Host → drive
DA2	36	Bit 2 bus địa chỉ thanh ghi trong	SA2	Host → drive
CS1Fx	37	Chọn chip cho địa chỉ cơ sở 1F0h	-	Host → drive
CS3Fx	38	Chọn chip cho địa chỉ cơ sở 3F0h	Host → drive	
DASP	39	Kích hoạt ổ đĩa/có ổ tớ	-	Drive → host
GND	40	Đất	-	-

(1) Tín hiệu đảo của tín hiệu bus AT

(2) Đầu được khoá để tránh cắm nhầm các chân nối.

(3) Tuỳ chọn

CPU thâm nhập bộ điều khiển IDE qua một vài thanh ghi số liệu và điều khiển... Chúng được phân thành hai nhóm với địa chỉ cơ sở của cảng ở 1F0h và 3F0h. Dưới đây liệt kê các thanh ghi đó.

Thanh ghi	Địa chỉ	Bề rộng thanh ghi	Đọc (R)/Viết(W)
Thanh ghi số liệu	1F0h	16	R/W
Thanh ghi lỗi	1F1h	8	R
Bù trừ trước	1F1h	8	W
Số các sectơ	1F2h	8	R/W
Địa chỉ sectơ	1F3h	8	R/W
Cylinder LSB	1F4h	8	R/W
Cylinder MSB	1F5h	8	R/W
Điều khiển ổ đĩa/ đầu từ	1F6h	8	R/W
Thanh ghi trạng thái	1F7h	8	R
Thanh ghi lệnh	1F7h	8	W
Thanh ghi trạng thái biến đổi	3F6h	8	R
Thanh ghi lối ra số	3F6h	8	W
Địa chỉ ổ đĩa	3F7h	8	R

Ví dụ, cấu trúc của thanh ghi điều khiển ổ đĩa/đầu từ tại địa chỉ 1F6h như sau:

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	DRV	HD_3	HD_2	HD_1	HD_0

DRV: ổ đĩa (1 = tớ; 0 = chủ);

HD0-DH3 : Số đầu từ (nhị phân) 0000 = đầu 0; 0001 = đầu 2... ; 1111 = đầu 15

Thí dụ, thanh ghi trạng thái tại 1F7h:

D7	D6	D5	D4	D3	D2	D1	D0
BSY	RDY	WFT	SKC	DRQ	CORR	IDX	ERR

BSY : bận	1 = ổ đĩa bận	0 = không bận
RDY : sẵn sàng	1 = ổ đĩa sẵn sàng	0 = không sẵn sàng
WFT : lỗi viết	1 = lỗi	0 = không lỗi
SKC : chỉnh vị trí đầu từ	1 = tìm xong	0 = đang tìm
DRQ : số liệu	1 = có thể truyền được	0 = không
CORR : lỗi số liệu sửa được	1 = có lỗi	0 = không lỗi
IDX : phát hiện tín hiệu vận tốc quay	1 = phát hiện	0 = không
ERR : lỗi	1 = thanh ghi lỗi đang chứa các thông tin về lỗi	0 = không chứa

Chương trình ghép nối CPU với IDE gồm ba giai đoạn:

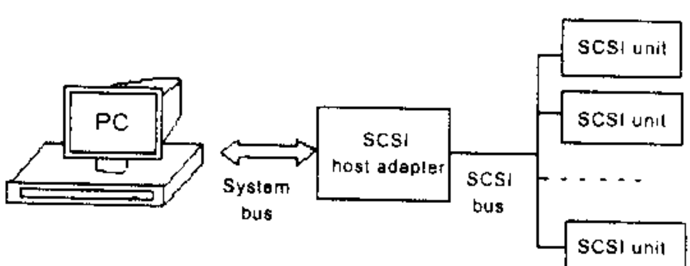
- Giai đoạn lệnh: CPU chuẩn bị tham số các thanh ghi và chuyển mã lệnh để khởi phát việc thực hiện lệnh

- Giai đoạn số liệu: Ổ đĩa định vị đầu từ và truyền số liệu giữa nó và bộ nhớ chính.

- Giai đoạn kết quả: Bộ điều khiển cung cấp thông tin trạng thái cho lệnh đã chạy trong các thanh ghi tương ứng và phát ra một ngắt qua IRQ14 (INT 76h)

Một chuẩn ghép nối nữa là chuẩn ghép nối SCSI (small computer system interface) có cấu trúc bus độc lập xuất hiện vào năm 1986. Khác với IDE chỉ là một thiết bị mở rộng bus hệ thống của PC, SCSI rất mềm dẻo và mạnh. Chuẩn này xây dựng một bus ghép nối gồm 8 đơn vị SCSI (SCSI unit) với bus hệ thống PC thông qua bộ giao diện (host adapter). Ngoài HDD có thể nối các thiết bị khác với chuẩn này như thiết bị băng từ, máy quét ảnh...

Hình 5-2 chỉ ra sơ đồ của bus SCSI.



Hình 5-2

Bộ giao diện (host-adapter) SCSI phức tạp và đắt tiền hơn giao diện IDE vì nó phải thực hiện các chức năng của bus SCSI, nhưng ngược lại nó lại không bị ràng buộc bởi những hạn chế của bus AT. Bus SCSI chỉ phục vụ cho việc trao đổi dữ liệu giữa những đơn vị được nối với nó. Trong một thời điểm chỉ có hai đơn vị có thể được kích hoạt. Việc trao đổi dữ liệu có thể được thực hiện giữa host adapter và ổ đĩa hoặc giữa hai đơn vị SCSI như ổ băng cassette và ổ đĩa mà không có sự tham gia của CPU vì các ổ đĩa SCSI đủ "thông minh" để tự làm điều đó. Giới hạn dung lượng ổ đĩa của SCSI là 8,4GB và tốc độ truyền dữ liệu đạt tới 5 Mbyte/s. Các phiên bản SCSI sau tốc độ lên tới 10MB/s và 40 MB/s với SCSI 32 bit.

Việc nối giữa các đơn vị SCSI được thực hiện bởi cáp dẫn bẹt 50 dây có các chân tín hiệu được chỉ ra như sau, cáp có cấu trúc xen kẽ dây tín hiệu và dây đất nên chống nhiễu tốt:

Tín hiệu	Chân cắm	Ý nghĩa	Tín hiệu	Chân cắm	Ý nghĩa
GND	1	Đất	GND	26	Đất nối
DB (0)	2	Bit số liệu 0	GND	27	Đất
GND	3	Đất	GND	28	Đất
DB (1)	4	Bit số liệu 1	GND	29	Đất
GND	5	Đất	GND	30	Đất
DB (2)	6	Bit số liệu 2	GND	31	Đất
GND	7	Đất	/ATN	32	Chú ý
DB (3)	8	Bit số liệu 3	GND	33	Đất
GND	9	Đất	GND	34	Đất
DB (4)	10	Bit số liệu 4	GND	35	Đất
GND	11	Đất	/BSY	36	Bận
DB (5)	12	Bit số liệu 5	GND	37	Đất
GND	13	Đất	/ACK	38	Ghi nhận
DB (6)	14	Bit số liệu 6	GND	39	Đất
GND	15	Đất	/RST	40	Reset
DB (7)	16	Bit số liệu 7	GND	41	Đất
GND	17	Đất	/MSG	42	Thông báo

DB (P)	18	Bit chẵn lẻ	GND	43	Đất
GND	19	Đất	/SEL	44	Chọn
GND	20	Đất	GND	45	Đất
GND	21	Đất	C/D	46	Lệnh/số liệu
GND	22	Đất	GND	47	Đất
GND	23	Đất	/REQ	48	Yêu cầu
GND	24	Đất	GND	49	Đất
(không nối)	25	-	I/O	50	Chiều bus

Thủ tục trao đổi thông tin như sau: Khi máy tính cần truy nhập ổ cứng nó đưa tín hiệu /SEL về 0. Ổ cứng trả lời bằng tín hiệu /ACK và /REQ. Tín hiệu C/D được dùng để phân biệt thông tin trên bus là lệnh (=1) hay dữ liệu (=0). Hướng truyền xác định được xác định bởi tín hiệu I/O, I/O=1 : dữ liệu vào ổ cứng, I/O=0 dữ liệu xuất từ ổ cứng. Khi mạch giao diện nhận xong dữ liệu, nó báo về ổ cứng bằng tín hiệu /ACK. Tín hiệu /BUSY báo bus SCSI đang hoạt động.

4. Tổ chức logic của đĩa từ

4.1. Sector logic

BIOS dùng các sector vật lí như trên để quản lý số liệu trong khi DOS và nhiều hệ điều hành khác lại dùng một sơ đồ khác gọi là sector logic. Đó là cách đánh số các sector vật lí một cách liên tục từ 0. Ví dụ cho một đĩa mềm hai mặt với 80 rãnh và 18 sector trên một rãnh, có dung lượng 1,44MB (kích thước 3,5 inch) thì có thể thấy rằng do có hai mặt nên trong một trụ (hai rãnh tương ứng ở hai mặt) có 36 sector đánh số từ 0 đến 35. Có 80 rãnh ở một mặt thành ra có 80 trụ, như vậy tổng số sector của đĩa mềm là 80x36=2880 và được đánh số từ 0 đến 2879 và phân bố như sau:

Sector vật lí		Sector logic
Head 0, track 0, sector 1		0
....	....	
Head 0, track 0, sector 18		17
Head 1, track 0, sector 1		18
....	....	
Head 1, track 0, sector 18		35
Head 0, track 1, sector 1		36
....	....	
....	....	
Head 1, track 79, sector 18		2879

4.2. Phân khu (Partition)

Một đĩa cứng có thể được chia thành nhiều ổ đĩa logic mà DOS sẽ có thể gán tên cho nó từ C đến Z và được đối xử như một ổ đĩa vật lí, đó là việc chia đĩa cứng thành các phân khu (partition). Về mặt logic, một phân khu có thể được coi như một đĩa cứng nên ta có thể cài một hệ điều hành tuỳ ý lên nó như MS-DOS, UNIX, OS/2...

Có ba loại phân khu trên đĩa cứng:

Phân khu DOS chính (primary DOS partition)

Phân khu DOS mở rộng (extended DOS partition)

Phân khu phi DOS (non-DOS partition)

Để lưu trữ thông tin về các phân khu. DOS thống nhất dùng một vùng cố định trong mọi đĩa cứng : head 0, rãnh 0, sector 1 và sector này cũng thường được gọi là sector phân khu (partitition sector). Thông tin về từng phân khu được lưu trữ bởi các điểm vào phân khu (partition entries) trong bảng phân khu (partition table)

Phần sau đây chỉ rõ cấu trúc của chúng:

Chương trình kiểm tra bảng phân khu và gọi sector khởi động (bootsector) (446 bytes)
Bảng phân khu (64 bytes)
Chữ kí (thường là AA55h) (2 bytes)

O ffice	Kích thước	Nội dung
1 beh (442)	16	partition 1
1 ceh (462)	16	partition 2
1 deh (478)	16	partition 3
1 eeh (494)	16	partition 4

Cấu trúc của một điểm vào phân khu

Office	Kích thước	Nội dung
00h	1	Chỉ thị boot 1,
01h-03h	3	Đầu phân khu
04h	1	Chỉ thị hệ thống 2
05h-07h	3	Cuối phân khu
08h-0hh	4	Sector khởi đầu 3
0ch-0fh	4	Số sector trong phân khu 3

Như vậy, bảng phân khu dài 64 byte, gồm cả 4 điểm vào (entry), mỗi điểm vào dài 16 byte lại được chia thành các vùng dữ liệu dài ngắn khác nhau chứa những thông tin cần thiết mô tả trọn vẹn một phân khu đĩa:

- Phân khu có thuộc loại tích cực (active) hay không, nghĩa là có thể có chức năng khởi động PC (boot) hay không.
- Số thứ tự đầu từ (head), sector, cylinder của nơi xuất phát phân khu.
- Phân khu được định dạng như thế nào, hoặc hệ điều hành nào được cài trên phân khu
- Số thứ tự head, cylinder của nơi kết thúc phân khu.

- Bao nhiêu sector nằm trước phân khu.
- Kích thước của phân khu tính theo số sector.

Chương trình FDISK.EXE trong DOS làm nhiệm vụ cung cấp thông tin và tạo các phân khu trên đĩa cứng.

4.3. Sector khởi động (boot record)

Vùng này dài 512 byte đặt tại sector logic 0 của ổ đĩa logic. Nó chứa một chương trình ngắn đặt biệt gọi là chương trình khởi động (boottrap loader) cho phép nạp DOS kernel lên RAM để khởi động PC. Bảng sau cho ta cấu trúc của một sector khởi động:

Mục	Vị trí	Nội dung	Số byte
1	00h	Chỉ thị nhảy, JUMP, về chương trình khởi động	3 byte
2	03h	Tên nhà sản xuất và số phiên bản	8 byte
3	0Bh	Số byte trên một sector	1 từ
4	0Dh	Số sector trên một cluster	1 byte
5	0Eh	Số sector dành cho boot sector	1 từ
6	10h	Số bản FAT (File Allocation Table)	1 byte
7	11h	Số điểm vào trên thư mục gốc (root directory)	1 từ
8	13h	Số sector trên tệp đĩa (volume)	1 từ
9	15h	Mô tả môi trường trữ tin (media descriptorr byte)	1 byte
10	16h	Số sector dành cho bản FAT	1 từ
11	18h	Số sector trên một track	1 từ
12	1Ah	Số đầu từ	1 từ
13	1Ch	Số sector ẩn	2 từ
14	1Eh	Số sector nếu kích thước lớn hơn 32Mb	2 từ
15	22h	Số ổ đĩa	1 byte
16	23h	Dự trữ	1 byte
17	24h	Chữ ký boot sector nới rộng	1 byte
18	25h	Số thứ tự	4 byte
19	29h	Tên tệp đĩa	11 byte
20	34h	Số nhận diện hệ thống tệp tin	8 byte
21	3Ch-200h	Chương trình khởi động (boot routine)	452 byte

Mục 9 (offset 15h = 21) mô tả môi trường trữ tin (media deseriptor byte) được đặt ở trong ba vùng trên đĩa:

sector 0, offset 21 trong boot record như trên

sector 1, offset 0 của bảng FAT 1

sector 1, offset 0 của bảng FAT 2

Nó xác định môi trường trữ tin thuộc loại nào: đĩa cứng, đĩa mềm, bao nhiêu mặt, bao nhiêu sector trên một track .v.v...

Mục 21 thường được gọi là bootstrap loader là chương trình khởi động viết theo ngôn ngữ máy, cũng có khi được gọi tắt là MBP (master boot program). Chương trình này lo việc nạp hệ điều hành DOS từ đĩa vào bộ nhớ RAM rồi trao quyền điều khiển cho DOS. Chương trình sẽ sử dụng các thông tin boot record để xác định loại đĩa từ nào (đĩa cứng hay mềm) mà DOS đang hoạt động.

4.4. Bảng định vị file: FAT

DOS lưu các file lên đĩa theo các đơn vị gọi là cluster. Mỗi cluster chứa một nhóm gồm một hoặc nhiều sector (thí dụ, trên đĩa cứng, một cluster gồm 4 hoặc 8 sector nghĩa là dài 2 Kbyte hoặc 4 Kbyte). Để theo dõi những cluster nào được sử dụng, cluster nào còn trống, DOS dựa trên một cấu trúc gọi là bảng FAT (file allocation table). DOS sẽ tìm kiếm file nằm ở đâu trên đĩa dựa vào bảng FAT sau khi tham khảo thư mục gốc.

Bảng FAT là một bảng gồm một số điểm vào ứng với số cluster có trên đĩa. Bề dài của mỗi điểm vào tùy thuộc phiên bản DOS, có thể là 12, 16 hay 32 bit, do đó có 2 loại bảng FAT-12, FAT-16 và FAT32. Các điểm vào chứa các mã cho biết các thông tin sau (ví dụ với FAT16):

- Hai điểm vào đầu tiên chứa mã để nhận dạng loại đĩa: F8FF FFFF là ổ cứng, FOFF FFFF là đĩa mềm.

- Các điểm vào sau cho biết:

Mã	ý nghĩa
0000h	cluster tương ứng còn trống, có thể ghi tin
FFF0h-FFF6h	cluster dành riêng
FFF7h	cluster tương ứng bị hỏng
FFF8h-FFFFh	file kết thúc và cluster tương ứng là cuối của file
xxxxh	cluster số xxxxh chứa phần kế tiếp của file

Với FAT 12 trong các mã bỏ đi một số 0, số F, số x vì một điểm vào chỉ có 12b tương đương với 3 số hệ 16. Với FAT32 thì lại phải thêm vào cho đủ 8 số hệ 16.

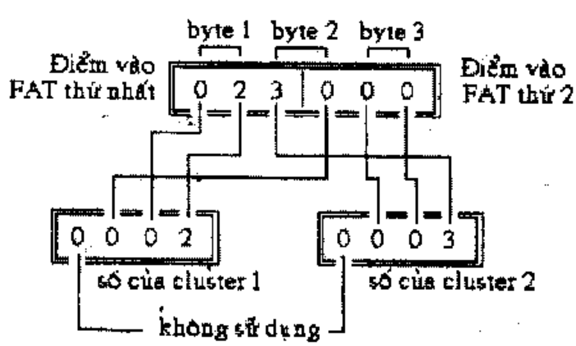
Với FAT-12, DOS l y 1,5 byte (từ 00h – FFFh) cho một điểm vào có thể biểu diễn được 4096 giá trị. Do vậy nó phải ghép các điểm vào FAT theo từng cặp hai điểm vào một để có 2 x 1,5 = 3 byte như trong hình 5-3.

Trước khi đi vào phân tích cách đọc bảng FAT ta hãy xem DOS lưu trữ các file như thế nào.

- Đầu tiên DOS xác định một điểm vào chưa sử dụng trong thư mục gốc và lưu vào đó tên file, thuộc tính file, ngày giờ tạo lập.

- DOS tìm trong bảng FAT điểm vào đầu tiên (theo thứ tự tính từ đầu bảng FAT) đánh dấu một cluster chưa sử dụng (mã 0h) và chứa số hiệu của cluster đó vào thư mục gốc. Ví dụ đó là điểm vào 9 đang chứa mã 0h, tức là số hiệu của cluster tương ứng là 9.

- Nếu dữ liệu chứa được trong một cluster, DOS chứa nó trong cluster 9 và viết mã FFFF vào điểm vào 9. Nếu dữ liệu của file hiện hành lớn hơn dung lượng một cluster thì DOS tiếp tục tìm cluster gần nhất chưa được sử dụng (có thể không liền kề với cluster đầu tiên). Ví dụ nó tìm thấy điểm vào Ah, nó sẽ lưu dữ liệu vào cluster Ah và viết mã 000A vào điểm vào 9. Quá trình cứ tiếp tục cho đến khi dữ liệu của file được lưu hết và điểm vào cuối cùng của file trong FAT sẽ chứa giá trị FFFFh.



Hình 5-3

Với FAT-16, DOS dùng 2byte để ghi thông tin cho một cluster với định dạng Intel (tức là đảo 2byte).

Ví dụ: Giải thích 16 byte của bảng FAT-16

Điểm vào 0 1 2 3 4 5 6 ... 06C3 ...

F8 FF FF FF 03 00 04 00 05 00 FF FF C3 06 ...11 DA

- F8FF: đĩa cứng

- FF FF: cluster kết thúc file nàođó, cluster khởi phát cho file là 2 được đọc từ thư mục gốc. DOS đầu tiên đọc file ở cluster 2.

- Để xác định cluster tiếp, phải đọc ở điểm vào cluster 2 thấy số: 03 00 đ (0003), nên đọc tiếp file ở cluster 3, truy nhập điểm vào 3 đọc được: 04 00 đ đọc tiếp tại (0004)... Ta nhận được chuỗi cluster 2đ3đ4đ5. Điểm vào thứ 5 có giá trị FFFF là kết thúc file. Vậy file này chứa trong 4 cluster. Tiếp theo, tra thư mục gốc thấy một file khác bắt đầu ở cluster 6, file sẽ tiếp tục ở cluster 06C3, DA11...

Có những phần mềm tiện ích như DISKEDIT trong Norton Utility cho ta ngay trị số của bảng FAT rất tiện dụng trong việc khảo sát, sửa chữa nó.

Với cách tổ chức bảng FAT để quản lý file như trên, các file không nhất thiết phải ghi lên các cluster liên tục kế cận nhau mà có thể nằm rải rác ở nhiều nơi, do vậy tiết kiệm được chỗ lưu trữ (hiện tượng phân mảnh – fragmentation). Khi xoá file chỉ cần cho điểm vào tương ứng bằng 0h để báo cho biết cluster đó là trống.

DOS sao ra hai bảng FAT giống hệt nhau để dùng cho mục đích bảo vệ. Hai bảng FAT này ban đầu được tạo ra do lệnh FORMAT. COM. Bảng 1 có thể hỏng trong quá trình truy nhập của hệ điều hành và nhiều nguyên nhân khác, khi đó có thể sao bảng 2 sang bảng 1.

Do dung lượng ổ đĩa cứng ngày càng lớn, kéo theo số cluster trên các phân khu hiện nay nhiều khi vượt quá 216 = 65536 là giới hạn mà bảng FAT-16 còn có thể quản lý được. Vì vậy thời gian gần đây bảng FAT-32 đã ra đời, trong đó thông tin cho một cluster được ghi lên 4 byte nên bảng này có thể quản lý được nhiều cluster hơn. Thêm nữa, kích thước mỗi cluster trong FAT-32 được quy định nhỏ hơn trong FAT-16 (4 Kbyte thay vì cho 32 Kbyte), do vậy nếu các file

dữ liệu không dùng hết cluster cuối cùng thì phần dư (phần lãng phí không được sử dụng) cũng sẽ nhỏ hơn. Điều này làm tăng đáng kể dung lượng đĩa còn trống. Nhưng tất nhiên số lần truy cập bảng FAT của DOS sẽ tăng do file chứa trong nhiều cluster hơn và điều này làm giảm tốc độ hệ thống. Có thể dễ dàng xác định được FAT32 có khả năng quản lý được ổ cứng dung lượng khoảng S= 2^{32}x4KB=16TB, vì FAT32 có 2^{32} điểm vào do vậy quản lý tối đa được 2^{32} cluster, dung lượng của một cluster lại bằng 4KB. Tương tự có thể tính được FAT16 quản lý được dung lượng tối đa S=2^{16}x32KB=2GB.

4.5. Thư mục gốc

Thư mục gốc (root directory) đi liền sau bảng FAT thứ hai. Đó là một bảng gồm nhiều điểm vào có bề dài 32 byte cho phép xác định ghi các tên file, thư mục đang lưu trữ trên đĩa và các thông tin khác về chúng. Bảng sau là nội dung của một điểm vào thư mục gốc

Vị trí	Nội dung	Kích thước
00h	Tên file (có điền kí tự trắng)	8 byte
08h	Phần phân loại (có điền kí tự trắng)	3 byte
0Bh	Thuộc tính file	1 byte
0Ch	Dự trữ	10 byte
16h	Giờ thay đổi thông tin cuối cùng	2 byte
18h	Ngày thay đổi thông tin cuối cùng	2 byte
1Ah	Cluster đầu tiên của file	2 byte
1Ch	Kích thước file	4 byte

4.6. Thâm nhập ổ đĩa

4.6.1. Thâm nhập ổ đĩa mềm

- Thâm nhập qua DOS

DOS cung cấp 3 ngắt cho việc thâm nhập ổ đĩa mềm và cứng là 25h, 26h và 21h. Ngắt 25h và 26h cho phép thâm nhập ở mức thấp khi có thể địa chỉ trực tiếp các sector logic. Dưới đây là các định dạng gọi và trở về cho ngắt INT 25h khi đọc và INT 26h khi viết các sector logic. Phân khu lên tới 32 Mbyte hoặc 65536 sector.

Bảng sau cho biết định dạng gọi và giá trị trả về (khi có lỗi) cho ngắt 25h (bên phải) khi đọc và 26h khi viết sector.

Thanh ghi	Giá trị gọi	Giá trị trả về
AL	Số ổ đĩa (1)	Mã lỗi xem bảng mã lỗi
AX		
CX	Số sector	
DX	Sector đầu	
BX	Offset và địa chỉ đoạn của vùng đệm đọc	
DS		

Thanh ghi	Giá trị gọi	Giá trị trả về
AL	ổ đĩa số (1)	Mã lỗi
AX		
CX	Số sector	
DX	Sector đầu	
BX	Offset và địa chỉ đoạn của vùng đệm viết	
DS		

(1) 0 = ổ A, 1 = ổ B...

Danh sách các mã lỗi cho INT 25h và INT 26h

Mã	Lỗi
01h	Lệnh không hợp lệ
02h	Che địa chỉ không chính xác
04h	Không tìm thấy sector
08h	Tràn trên DMA
10h	Lỗi CRC hoặc ECC
20h	Lỗi bộ điều khiển
40h	Lỗi tìm kiếm
80h	Ổ đĩa không sẵn sàng

Với các hàm trong ngắt 21h, DOS cung cấp những dịch vụ tìm, đọc và viết các sector, cập nhật ngày tháng và kích thước file...

- Thâm nhập qua BIOS

Việc thâm nhập ổ đĩa được thực hiện qua ngắt INT 13h, với 6 hàm từ 00h đến 05h.

Nội dung của các hàm (chức năng) như sau:

00h	Khởi động ổ đĩa
01h	Đọc trạng thái và tác vụ cuối cùng
02h	Đọc các sector
03h	Viết các sector
04h	Kiểm tra các sector
05h	Định dạng track

Ví dụ, đọc sector 1 của track 0, head 0 của đĩa mềm trong ổ đĩa B (sector này chứa sector khởi động của đĩa):

```
buffer   DB 512 DUP        ; cấp vùng đệm dọc 512 bytes (=1 sector)
MOV      AH,02h            ; gọi hàm 02h (đọc sector)
MOV      DL,01h            ; ổ đĩa mềm thứ 2
MOV      DH,00h            ; đầu từ 0
MOV      CH,00h            ;track 0
MOV      CL,01h            ; sector 1
MOV      AL,01h            ; chỉ đọc 1 sector
MOV      ES, SEG buffer    ; địa chỉ đoạn vùng đệm dọc
MOV      BX, offset buffer ; địa chỉ offsect vùng đệm dọc
INT 13h                    ; đọc sector vào vùng đệm bằng ngắt 13h
```

4.6.2. Thâm nhập đĩa cứng qua DOS và BIOS

Giống như với ổ đĩa mềm, các sector logic có thể được thâm nhập qua ngắt 25h và 26h của DOS. Các sector vật lí được thâm nhập qua ngắt 13h của BIOS. Các hàm trên 04h dùng cho đĩa cứng như sau:

Dưới đây là danh sách các mã lỗi điều khiển:

05h	Định dạng track và Cylinder
06h	Định dạng và đánh dấu track xấu
07h	Định dạng và đánh dấu ổ đĩa
08h	Xác định các thông số ổ đĩa
09h	Đặt các thông số cho ổ đĩa

0Ah	Đọc các sector mở rộng
0Bh	Viết các sector mở rộng
0Ch	Tìm kiếm (seek)
0Dh	Khởi động đĩa cứng
0Eh	Đọc bộ đệm sector
0Fh	Viết bộ đệm sector
10h	Kiểm tra ổ đĩa đã sẵn sàng chưa
11h	Chuẩn lại ổ đĩa
19h	Nâng đầu từ đọc/viết

Dưới đây là danh sách các mã lỗi điều khiển:

Mã	Mô tả	Mã	Mô tả
00h	Không có lỗi	19h	Rãnh xấu không có rãnh biến đổi
02h	Không có tín hiệu tìm kiếm	1Ah	Số liệu đọc lỗi
03h	Lỗi viết	1Bh	Số liệu viết lỗi
04h	Ổ đĩa không sẵn sàng	1Ch	Track biến đổi không bị che
06h	Không tìm thấy track 0	1Dh	Lỗi tuần tự
10h	Lỗi ECC trong trường ID	1Eh	Th/nhập bất hợp lệ tới track đổi
11h	Lỗi ECC trong trường số liệu	20h	Mã toán bất hợp lệ
12h	Không có che địa chỉ ID	21h	Địa chỉ khối logic bất hợp lệ
13h	Không có che địa chỉ số liệu	22h	Thông số bất hợp lệ
14h	Không có trường ID	23h	Tràn
15h	Lỗi tìm kiếm (seek)	30h	Lỗi trong vùng đệm sector
16h	Lỗi bộ điều khiển bên trong	31h	Lỗi kiểm tra ROM
17h	Lỗi DMA	33h	Lỗi bên trong CPU
18h	Lỗi số liệu có thể sửa được		

II. ĐĨA QUANG

1. Nguyên tắc làm việc

Tương tự như đĩa từ, đĩa quang là một môi trường lưu trữ dữ liệu ngay cả khi mất nguồn điện. Thông tin được lưu trữ trên đĩa quang dưới dạng thay đổi

tính chất quang trên bề mặt đĩa và làm thay đổi chất lượng phản xạ một tia sáng laser (bước sóng cố định trong dải 790-850nm gần vùng hồng ngoại) trên bề mặt đĩa. Tia laser được hội tụ vào một điểm rất nhỏ cỡ vài phần mười μm, vì thế đĩa quang có dung tích lưu trữ lớn hơn nhiều so với đĩa từ. Nhưng nhược điểm chính là tốc độ đọc chậm hơn đĩa từ. Thời gian truy nhập một đĩa CD-ROM nhỏ hơn 150ms, còn của DVD cỡ 100ms. Tốc độ đọc cơ sở (1X) của đĩa CD là 150 Kbyte/s, còn của DVD là 1,321 Mbyte/s, hiện nay đã có những ổ tốc độ 52X (52x150KB với CD).

Đĩa quang được chia thành 4 loại chính : CDROM (compact disk) là loại tin đã được ghi sẵn khi sản xuất đĩa. CD-R (reccordable compact disk) hay đĩa WORM (write once read many), loại này người dùng có thể tự ghi được. CD-WR (writeable/readable compact disk) là loại viết được nhiều lần. Đĩa DVD (digital video disk) có mật độ ghi tin cao. Cuối cùng là các đĩa quang từ có thể viết được nhiều lần nhưng giá thành cao nên không cạnh tranh được với các loại đĩa trên.

Các đĩa quang thường áp dụng nguyên tắc mã hoá RLL, tuy nhiên do thông tin trên đĩa quang dễ bị nhiễu ví dụ như khi có bụi nằm giữa tia laser và điểm đọc, do vậy cần kiểm tra, sửa lỗi nhiều hơn đĩa từ và vì thế cần nhiều thông tin ECC hơn đĩa từ.

Trong ổ đĩa quang dùng laser bán dẫn phát ra từ vùng tiếp xúc p-n để đọc và ghi đĩa quang, công suất khoảng 5mW, không ảnh hưởng đến mắt người. Trong ổ đĩa thường có hai điôt cảm quang, một điôt đọc để biến tín hiệu quang thành tín hiệu điện để xử lý tiếp, điôt thứ hai dùng để kiểm tra cường độ tia laser để hiệu chỉnh nếu công suất phát sáng suy giảm theo thời gian.

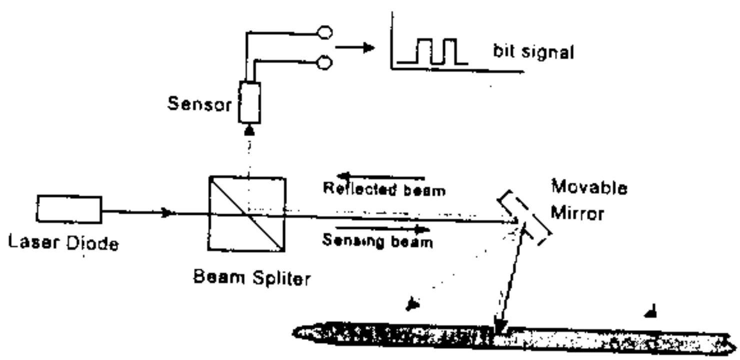
Rãnh dữ liệu của đĩa quang là đường xoắn ốc liên tục nên đĩa quang cần quay trong chế độ vận tốc tuyến tính không đổi (đĩa từ làm việc trong chế độ vận tốc góc không đổi). Theo chuẩn CDROM vận tốc tuyến tính của đĩa là 1,3 m/s, để giữ vận tốc này không đổi, vận tốc quay của đĩa cần thay đổi từ 500 vòng/s ở bên trong đến 200 vòng/s ở bên ngoài. Phương pháp hiệu chỉnh tốc độ tuyến tính tương đối đơn giản, bộ đệm dữ liệu luôn được giữ đầy 50%, nếu lượng dữ liệu nhỏ hơn 50% bộ đệm, tốc độ động cơ được tăng lên. Với ổ CD tốc độ 32X dữ liệu được đọc ra từ bộ đệm của ổ CDROM với tốc độ 4,32 MB/s. Tuy nhiên việc phải điều chỉnh tốc độ động cơ làm cho thời gian tìm dữ liệu lâu hơn.

Để chỉnh cho đầu đọc luôn vào đúng rãnh khi khoảng cách giữa hai rãnh gần nhau nhất ,ví dụ với CDROM là 1,6mm cần một bộ phận chỉnh vị đặc biệt. Khi ấy có ba tia laser: một tia chính và 2 tia phụ hai bên. Hai tia phụ đi theo hai tia chính và gặp 2 điôt cảm quang E, F nằm hai bên điôt chính. Nếu tia chính nằm đúng rãnh, tín hiệu hai tia phụ bằng nhau và hiệu số của chúng bằng 0, nếu tia chính lệch khỏi rãnh, một tia phụ sẽ gặp rãnh thông tin nhiều hơn, do đó khi lệch trái, ví dụ E-F>0 thì khi lệch phải E-F<0. Tín hiệu từ hai điốt cảm quang sẽ khác nhau và tín hiệu từ bộ khuếch đại vi sai được dùng làm tín hiệu chỉnh vị tia laser đọc.

2. Cấu trúc vật lý

2.1. Đĩa CD: Đĩa CD dùng trong PC có dung lượng điển hình cỡ hơn 500 Mbyte với tốc độ truyền số liệu cơ sở (1X) cỡ 150Kbyte/s và thời gian thâm nhập đĩa chưa cao lắm cỡ nhỏ hơn 150ms.

Nguyên tắc hoạt động của CD-ROM gần như máy CD thông thường trong kỹ thuật âm thanh (audio) như hình 5-4.



Hình 5-4

Đĩa được ghi các thông tin ở dạng những lỗ nhỏ (pit) đường kính 0,6μm và sâu 0,12μm trên mặt đĩa. Các rãnh cách nhau 1,6μm, mật độ rãnh là 16000 tpi (track per inch). Cường độ tia phản xạ sẽ yếu đi khi gặp các lỗ nhỏ. Trong ổ đĩa, có một sensor thu, nhạy với cường độ tia phản xạ. Cường độ tia phụ thuộc vào việc nó đi qua lỗ hay mặt phẳng (land), do vậy phụ thuộc vào các bit thông tin được ghi trên đĩa. Lối ra của sensor sẽ là các tín hiệu điện phù hợp với các bit số liệu được ghi trên đĩa.

Đĩa CDROM loại đường kính 120mm, dày 1,2mm có 22188rãnh. Dữ liệu

chia thành từng khối như trong đĩa từ (cung hay liên cung). Mỗi khối bao gồm: 12B đồng bộ, 4B địa chỉ khối, 2048B dữ liệu và 288B mã sửa lỗi. Dữ liệu được lưu trữ theo mã RLL2,7.

2.2. Đĩa CD-R: Là đĩa CD có thể viết được bởi người sử dụng (giống PROM), đĩa này có thể sử dụng trong bất cứ ổ CD nào. Nó gồm các lớp sau: lớp phủ chống xước, lớp phim bảo vệ tia tử ngoại, lớp phim phản xạ bằng vàng hay hợp kim bạc dày cỡ 100 nm, lớp màu polymer hữu cơ là lớp chứa dữ liệu và trong cùng là lớp polycarbonat trong suốt. Ngoài tia laser đọc ra, ổ đĩa này còn có một tia laser gọi là tia viết có cường độ mạnh hơn nhiều để tạo ra một xung đốt cháy lớp màu trong quá trình ghi thông tin lên đĩa. Lớp màu này chuyển sang màu đen và đóng vai trò các lỗ dữ liệu của CD thường. Có hai cách ghi CD-R: ghi một lần toàn đĩa và ghi từng rãnh một lần (ghi đĩa nhiều lần) ít nhất 300 khối.

2.3. Đĩa CD-RW: Là đĩa ghi lại được, bao gồm các lớp sau: lớp phủ chống xước, lớp phim phản xạ bằng vàng hay hợp kim bạc dày cỡ 100nm, lớp cách điện trên, lớp kim loại lưu trữ dữ liệu, lớp cách điện dưới, lớp polycarbonat trong suốt. Nguyên tắc ghi dữ liệu dựa theo sự thay đổi trạng thái của lớp kim loại: trạng thái tinh thể (phản xạ ánh sáng tương đương vùng trống-land) và trạng thái vô định hình (không phản xạ ánh sáng tương đương vùng lỗ-pit). Quá trình thay đổi trạng thái này có thể thay đổi bất kỳ tuỳ theo công suất laser, vì thế CD-RW có thể xoá đi và ghi lại được. Để thực hiện điều này, ổ đĩa CD-RW sử dụng 3 mức công suất laser khác nhau: công suất cao hay công suất ghi để tạo trạng thái vô định hình, công suất vừa hay công suất xoá để tạo trạng thái tinh thể và công suất thấp để đọc dữ liệu. Phần lớn các ổ CDROM và CD-R không đọc được CD-RW vì độ phản xạ kém của CD-RW.

2.4. Đĩa quang mật độ cao DVD: Loại đĩa này sẽ thay thế các loại đĩa CD trong tương lai vì có dung lượng lớn hơn và khả năng truy nhập nhanh hơn. Để đạt được điều đó, trước hết DVD có kích thước lỗ nhỏ hơn cỡ 2 lần, khoảng cách giữa các rãnh gần hơn ... nên với một lớp dữ liệu dung lượng DVD lớn hơn CD cỡ 7 lần. Hơn nữa DVD có thể dùng nhiều lớp dữ liệu để lưu trữ tin, khoảng cách giữa hai lớp cỡ 20-70 μm và để chống nhiễu thì lỗ ở lớp 2 lớn hơn lớp 1 và tốc độ đọc lớp 2 nhanh hơn lớp 1. Tốc độ truy nhập cơ bản (1X) của ổ DVD là 1,321MB/s, thời gian truy nhập cỡ 100ms. Nói chung tốc độ truy nhập được đánh giá cao hơn CD cỡ 9 lần. Các ổ DVD4X đọc được CDROM với tốc độ

32X. Giao diện nối ổ DVD với máy tính giống ổ CDROM. DVD-R và DVD-RW có dung lượng trung bình cỡ 4,7GB.

2.5. Đĩa quang từ: Nguyên tắc của nó là sử dụng ảnh hưởng của từ trường lên sự phân cực của sóng điện từ.

Ánh sáng của tia laser là sóng điện từ được phân cực theo một hướng xác định. Khi sóng điện từ đi qua một vật liệu đặt trong từ trường, từ trường đó sẽ tác dụng lên trường điện từ của sóng và làm quay mặt phẳng phân cực của nó, đó là hiệu ứng Faraday mà chúng ta đã biết. Đĩa quang – từ sử dụng hiệu ứng đó để ghi thông tin. Đĩa có một lớp phủ đồng nhất làm bằng vật liệu hợp kim lantan sắt từ. Khi tia laser phân cực đốt bề mặt đĩa, hướng phân cực của tia phản xạ được quay phụ thuộc vào từ độ của bề mặt. Một bộ lọc phân cực được dùng làm kính phân tích (analyzer) cho hướng phân cực của tia phản xạ và sensor sau bộ lọc nhạy với cường độ của ánh sáng đi qua bộ lọc phân cực đó. Khi cần viết một bit lên đĩa, một xung laser ngắn và mạnh sẽ đốt bề mặt đĩa ở vùng cần thiết lên quá nhiệt độ Curie Tc làm cho từ độ của vùng phủ sắt từ đó bằng 0. Cùng lúc đó, một nam châm phát ra từ trường với hướng phụ thuộc vào giá trị của bit cần viết ("1" sẽ có hướng ngược với "0"). Hướng của từ trường này bây giờ xác định hướng của các domain từ trong vùng bị đốt nóng khi mà chúng được làm lạnh xuống dưới nhiệt độ Curie. Khi lớp phủ đó được đông rắn lại, quá trình viết đã hoàn thiện. Khi đọc một bit, tia laser sẽ quét bề mặt đĩa và hệ phân cực sẽ nhạy với hướng phân cực tia phản xạ. Nếu chiếu tia laser trên vùng viết bit, hướng của mặt phẳng phân cực tia phản xạ sẽ phụ thuộc vào hướng của từ độ của lớp phủ bề mặt vùng đó. Ví dụ, hướng phân cực trái sẽ đại diện cho mức logic "0" và phân cực phải cho mức "1". Trong trường hợp thứ nhất, ánh sáng không thể đi tới bộ lọc phân cực được và sensor cho tín hiệu điện tương ứng với giá trị 0. Trong trường hợp sau, tia sáng sẽ đi qua bộ lọc tới sensor với cường độ sáng toàn bộ và sensor cho tín hiệu điện tương ứng với giá trị 1. Bằng cách làm nóng các điểm xác định với hướng phù hợp của từ trường được phát bởi nam châm, bit được viết có thể được xoá. Việc xử lý viết có thể làm đi làm lại nhiều lần như đĩa từ.

Nhược điểm của đĩa quang từ là dễ bị gỉ nên bề mặt đĩa phải phủ một lớp nhựa hay thuỷ tinh, do đó giá thành đắt nên không cạnh tranh được với đĩa CD-RW và DVD-RW.

3. Cấu trúc logic

Về nguyên tắc đĩa CD có thể chứa các tệp của hệ điều hành như đĩa cứng. Đĩa CD nào cũng có một vùng mục lục TOC (table of content). Vùng này xác định vị trí bắt đầu và chiều dài của rãnh dữ liệu trên đĩa. Không có TOC, sẽ không đọc được đĩa. CDROM thường dùng hệ file chuẩn ISO9660. Phiên bản đầu tiên là ISO9660.1 định nghĩa tên file theo khổ 8+3 như FAT của DOS, số lớp thư mục tối đa là 8, một file bao gồm nhiều khối liên tục... Phiên bản ISO9660.3 cho phép một file gồm nhiều khối rời rạc. Hệ điều hành cần một chương trình biên dịch hệ file ISO9660 thành hệ file của hệ điều hành thì mới quản lý được CD.

Bài tập chương 5

1.Tính dung lượng ổ cứng mà FAT 32 quản lý được. Cho độ dài liên cung là 32KB.

2.Tính dung lượng ổ cứng mà FAT 16 quản lý được với độ dài liên cung lần lượt là 32 KB và 4KB.

3.Tại sao nói dùng liên cung nhỏ tiết kiệm không gian đĩa hơn. Khi đó lại mắc phải nhược điểm gì.

Chương 6

THIẾT BỊ NGOẠI VI VÀ GHÉP NỐI

Mục tiêu: Như đã trình bày trong chương 1, thiết bị ngoại vi được nối vào máy tính thông qua card giao diện. Card giao diện có vị trí nằm giữa máy tính và thiết bị ngoại vi, có vai trò trung chuyển thông tin giữa hai thiết bị này. Card giao diện và thiết bị ngoại vi tạo thành hệ thống thiết bị ngoại vi. Máy tính chỉ sử dụng tín hiệu số với mức lôgic duy nhất là TTL (mức 1=5v, mức 0 = 0v), trong khi đó thiết bị ngoại vi có thể sử dụng các dạng tin khác nhau : tương tự, số, chữ số, âm tần... Dạng tin số trao đổi của thiết bị ngoại vi cũng rất phong phú : trao đổi tin song song nhiều bit trên nhiều đường dây, trao đổi tin nối tiếp trên một đường dây hoặc trao đổi tin cả gói. Do vậy, card giao diện phải làm cả nhiệm vụ phối hợp dạng tin truyền giữa máy tính và thiết bị ngoại vi. Ví dụ, card giao diện cho các thiết bị truyền tin nối tiếp (chuột, bàn phím) nằm ngay trên bản mạch chính của máy tính, có nhiệm vụ biến đổi tin nối tiếp nhận được thành tin song song truyền lên bus máy tính. Thiết bị ngoại vi truyền tin nối tiếp cắm vào card giao diện qua giắc cổng COM, PS/2, giắc bàn phím, thiết bị ngoại vi trao đổi tin song song 8 bit cắm vào giắc LPT. Các thiết bị khác cắm vào card giao diện chế tạo theo chuẩn AGP hoặc PCI và chính các card này cắm vào các khe cắm mở rộng tương ứng trên bản mạch chính. Hiện nay đã xuất hiện các thiết bị ngoại vi sử dụng chuẩn USB, các thiết bị này trao đổi tin cả gói và được nối vào máy tính qua card chuẩn USB.

Mỗi một thiết bị ngoại vi và card giao diện kèm theo có một địa chỉ xác định, ví dụ chuột cắm qua cổng COM có địa chỉ 3F8h - 3FFh, máy in nối vào cổng LPT có địa chỉ 378h - 37Fh, bàn phím có địa chỉ 60h. Vi xử lý sẽ truy nhập địa chỉ này để trao đổi tin với thiết bị ngoại vi. Vi xử lý khi làm việc với mỗi thiết bị ngoại vi đều sử dụng một chương trình ngắt riêng, được gọi bằng một lệnh ngắt INT xx, trong đó xx là số hiệu ngắt đặc trưng cho từng thiết bị. Tóm lại mục tiêu của chương này trang bị cho sinh viên các kiến thức về cấu

trúc và nguyên tắc hoạt động cơ bản của một số thiết bị ngoại vi quan trọng; các thông tin về địa chỉ và chỉ số ngắt của các thiết bị này cũng như nguyên tắc ghép nối với máy tính.

I. MÁY IN VÀ GHÉP NỐI SONG SONG

Máy in là thiết bị đưa tin ra và lưu trữ trên giấy. Dưới đây trình bày nguyên tắc làm việc của hai loại máy in chính.

1. Máy in ma trận chấm

Nguyên tắc tạo chữ của máy in này cũng tương tự cách vẽ chữ hay ký hiệu trên màn hình tức một chữ tạo bởi nhiều chấm mức bố trí theo một ma trận. Do đó có thể thay đổi kích thước, kiểu chữ bằng chương trình.

Tuỳ theo cách tạo chấm, tức đầu in ta có các loại sau:

1.1. Máy in ma trận kim dùng mực

Đầu in gồm một số kim được điều khiển bởi các búa gõ điện từ. Khi búa gõ, băng mức được ấn vào giấy, một chấm mức được in.

Tuỳ theo số lượng kim và và kích thước búa gõ điện từ có thể bố trí các kim theo một hàng (in chậm) hay theo ma trận (ví dụ 14 hàng x 9 cột) để gõ đồng thời các búa gõ (in nhanh). Các mô tơ bước điều khiển đầu in và quay giấy. Khi bố trí kim theo hàng, thì phải quay giấy để quét hết ma trận chữ như in ma trận. Thường có các loại máy in 9, 25 kim trong một hàng.

Ngoài in chữ, có thể in được hình vẽ nhiều màu (phụ thuộc số băng mực màu) vì mỗi lần chỉ in một màu. Tốc độ in chấm ma trận-khoảng 50 - 350 chữ/giây.

Phông chữ được nạp cứng trong một ROM ký tự hoặc trong một file phông chữ (có nhiều loại phông chữ cho ta tuỳ chọn). Khi vi xử lý truyền đến ROM ký tự mã ASCII của ký tự cần in, một mẫu ký tự tương ứng được chọn từ ROM và đưa vào ma trận RAM để in. Còn khi in đồ hoạ thì mẫu bức tranh thể hiện bằng các bit 0, 1(ví dụ bit 1 tương đương động tác đẩy kim in ra hay tương đương một chấm mực) sẽ được đưa vào bộ đệm RAM. Trong trường hợp này không cần ROM ký tự vì mẫu ký tự cũng được coi là một bức tranh lấy từ file phông chữ

1.2. Máy in ma trận kim dùng nhiệt

Giống máy in dùng băng mực, nhưng thay cho búa cơ điện và băng mực

bằng các đầu đốt nóng đặt thành ma trận. Khi các kim này áp vào băng tác dụng nhiệt, khi băng bị nung nóng, vết nung nóng đó sẽ truyền sang mặt giấy bình thường thành vết đen do bị đốt cháy.

1.3. Máy in phun mực

Tia mực được phun ra giống như tia điện tử của màn hình và được tích điện. Khi đi qua tụ tích điện lái tia, tia bị lái và đập vào giấy. Điều khiển di chuyển đầu in, ta được ma trận chấm của các ký tự và pixel bằng mực lên giấy. Có hai loại phun:

- Có nhiều lỗ phun theo chiều dọc và in từng dòng điểm của ma trận và di chuyển đầu in theo chiều ngang, ta in được ma trận điểm.
- Có một lỗ phun, di chuyển đầu in theo chiều ngang (giống quét dòng của màn hình) và di chuyển giấy theo chiều dọc (giống quét mành của màn hình) nhưng tốc độ chậm hơn. Máy có độ phân giải 300 chấm/ inch tương đương máy in laser nhưng hơi mờ hơn vì giọt mực nhỏ.

2. Máy in laser

Nguyên tắc in bằng tia sáng laser như sau:

- Chùm tia sáng mỏng đi qua bộ điều chế tới mặt một trống phủ một lớp nhạy quang bằng selen để tạo nên hình ảnh bằng điện tích.
- Mực dạng bột đã tích điện được hút và bám vào mặt trống mà độ đậm nhạt tuỳ cường độ sáng đã chiếu.
- Bột mực sẽ bám vào giấy tích điện với điện thế cao hơn trống và tan ra trên giấy tạo hình ảnh ngược của trống nhạy quang khi qua một trống sấy nóng ở nhiệt độ cao.

Hiện nay máy in laser có thể in từ 10-12 trang /phút với độ phân giải 1200-2400 pixel/inch.

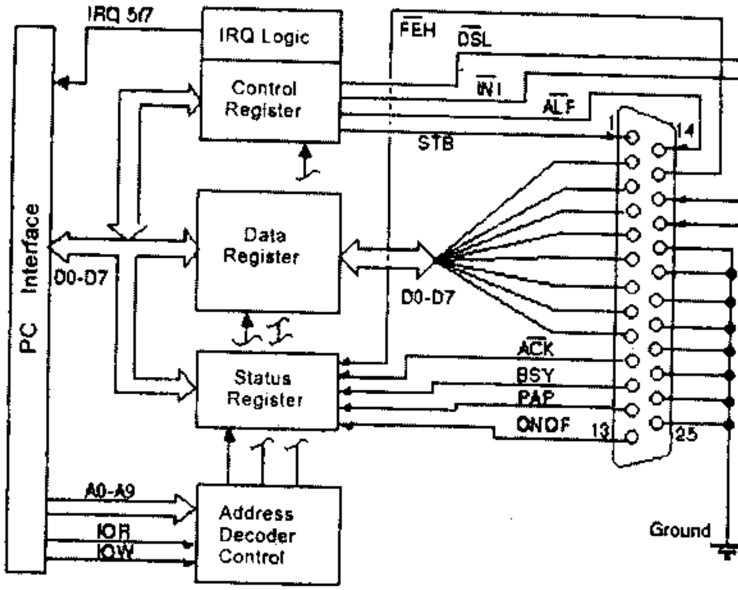
3. Ghép nối song song

Các máy PC được trang bị ít nhất là một cổng ghép nối song song và một cổng ghép nối nối tiếp. Khác với ghép nối nối tiếp có nhiều ứng dụng, ghép nối song song thường chỉ phục vụ cho ghép máy in với PC.

Sơ đồ khối của ghép nối song song như hình 6-1 sau:

Có ba thanh ghi có thể truyền số liệu và điều khiển máy in và khối ghép nối. Địa chỉ cơ sở của các thanh ghi cho tất cả cổng LPT (Line Printer) từ LPT1 đến LPT4 được lưu giữ trong vùng số liệu BIOS. Thanh ghi số liệu được định vị ở

offset 00h, thanh ghi trạng thái ở 01h và thanh ghi điều khiển ở 02h. Nói chung địa chỉ cơ sở của LPT1 là 378h và của LPT2 là 278h, do đó địa chỉ của thanh ghi trạng thái là 379h hoặc 279h và địa chỉ thanh ghi điều khiển là 37Ah hoặc 27Ah.



Hình 6-01

Dưới đây là định dạng các thanh ghi:

Thanh ghi số liệu (hai chiều) : Các bit số liệu từ D0 đến D7

7							0	
D7	D6	D5	D4	D3	D2	D1	D0	Tín hiệu máy in
9	8	7	6	5	4	3	2	Số chân cắm

Thanh ghi trạng thái máy in (chỉ đọc) :

7							0	
BSY	/ACK	PAP	OFON	/FEF	X	X	X	Tín hiệu máy in
11	10	12	13	15	-	-	-	Số chân cắm

Thanh ghi điều khiển máy in:

7							0	
x	x	x	IRQ	/DSL	/INI	/ALF	STR	Tín hiệu máy in
-	-	-	-	17	16	14	1	Số chân cắm

x : Không sử dụng (thường để ở mức 1)

IRQ, yêu cầu ngắt cứng : 1 = cho phép; 0 = không cho phép

Bản mạch ghép nối chỉ có bus số liệu 8 bit, điều này là phù hợp vì số liệu luôn đi qua máy in thành từng khối 8 bit một. Các chân tín hiệu của đầu cắm 25 chân của cảng song song LPT được liệt kê dưới đây:

Chân	**Tín hiệu**	**Mô tả**
1	$\overline{\text{STR}}$	Mức tín hiệu thấp truyền số liệu tới máy in
2	D0	Bit số liệu 0
3	D1	Bit số liệu 1
4	D2	Bit số liệu 2
5	D3	Bit số liệu 3
6	D4	Bit số liệu 4
7	D5	Bit số liệu 5
8	D6	Bit số liệu 6
9	D7	Bit số liệu 7
10	$\overline{\text{ACK}}$	Mức thấp chỉ rằng máy in đã nhận một kí tự và có khả năng nhận nữa.
11	BSY	Mức cao chỉ rằng: - kí tự đã được nhận - đầy bộ đệm máy in - khởi động máy in - máy in ở trạng thái off-line
12	PAP	Mức cao chỉ thị hết giấy.
13	OFON	Mức cao chỉ rằng máy in ở trạng thái on-line.

14	$\overline{ALF}$	Tự động xuống dòng; mức thấp chỉ thị rằng máy in xuống dòng tự động
15	$\overline{FEH}$	Mức thấp chỉ thị : - hết giấy - máy in ở off-line - lỗi máy in.
16	$\overline{INI}$	Mức thấp khởi động máy in.
17	$\overline{DSL}$	Mức thấp chọn máy in.
18 - 25	GROUND	Nối đất (0 Volt).

Thường tốc độ xử lí dữ liệu của các ngoại vi như máy in chậm hơn PC nhiều nên các đường ACK, BSY và $\overline{STR}$ được sử dụng cho kĩ thuật móc nối (hand-shaking). Khởi đầu PC đặt các số liệu lên bus sau đó kích hoạt đường STR xuống mức thấp để thông tin cho máy in biết rằng số liệu đã ổn định trên bus. Khi máy in xử lí xong dữ liệu, nó sẽ trả lại tín hiệu xuống mức thấp để ghi nhận. PC đợi cho đến khi đường BSY từ máy in xuống thấp (máy in không bận) thì sẽ đưa tiếp số liệu lên bus.

Số liệu có thể trao đổi trực tiếp giữa hai PC qua các cảng song song với nhau: Muốn vậy, các đường điều khiển bên này phải được nối với các đường trạng thái bên kia.

Máy in có thể được thâm nhập bằng chương trình qua DOS, BIOS hoặc trực tiếp qua các cảng được nêu ở trên. Các lệnh như "COPY tên - file PRN" trong DOS cho phép in một file ra máy in. Ngắt 17h với các hàm 01h khởi động máy in, 00h in một kí tự ra máy in, 02h trả về trạng thái của máy in v.v... có sẵn trong BIOS.

II. GHÉP NỐI NỐI TIẾP

1. Truyền số liệu nối tiếp và không đồng bộ

1.1. Truyền đồng bộ và không đồng bộ

Ghép nối nối tiếp (serial interface) cho phép trao đổi thông tin giữa các thiết bị từng bit một. Dữ liệu thường được gửi theo các nhóm bit SDU (Serial Data Unit). Các thiết bị ngoại vi như máy vẽ, modem, chuột và máy in có thể được nối với PC qua cảng nối tiếp COM.

Sự khác nhau giữa truyền dữ liệu nối tiếp đồng bộ và không đồng bộ chỉ là: trong kĩ thuật truyền đồng bộ, ngoài đường dây dữ liệu phải đưa thêm vào một đường tín hiệu đồng bộ để chỉ thị rằng khi nào bit tiếp theo ổn định trên đường số liệu. Ngược lại trong truyền không đồng bộ, các bit dữ liệu tự nó chứa các thông tin để đồng bộ: phần phát (transmiter) và phần thu (receiver) phải hoạt động với cùng một tần số nhịp đồng hồ. Thông tin đồng bộ (trong truyền không đồng bộ) gồm có các bit start (chỉ thị bắt đầu của khối dữ liệu được truyền - ví dụ bắt đầu của một byte) và một bit stop (chỉ thị kết thúc khối dữ liệu đó).

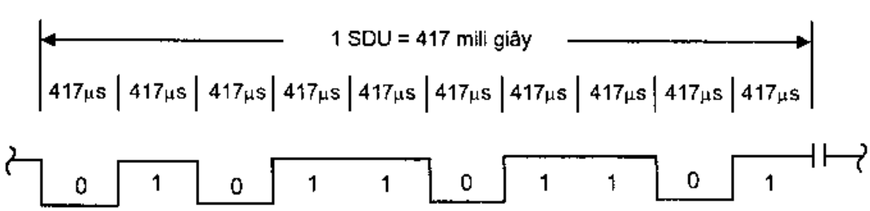
1.2. Kiểm tra chẵn lẻ và tốc độ truyền

Bit chẵn lẻ (parity bit) được đưa thêm vào khung SDU dùng để phát hiện lỗi trên đường truyền. Việc truyền bit chẵn lẻ chỉ kiểm soát được các lỗi trên đường truyền ngắn và các lỗi bit đơn, nên trong một số ứng dụng đặc biệt người ta phải dùng mã CRC mặc dù có phức tạp hơn. Dầu sao gần như tất cả các chip bổ trợ cho ghép nối nối tiếp ngày nay đều được thiết kế phần cứng kiểm tra chẵn lẻ.

Một thông số khác liên quan tới truyền dữ liệu nối tiếp là tốc độ truyền dữ liệu được gọi là tốc độ baud. Trong việc truyền mã nhị phân, đó là số bit được truyền trong một giây bps (bit per second).

1.3. Nhóm dữ liệu nối tiếp SDU và nối tiếp hóa

Trước khi truyền chuỗi số liệu nối tiếp, máy phát và máy thu phải được khởi tạo để hoạt động với cùng một định dạng dữ liệu, cùng một tốc độ truyền. Hình 6-2 chỉ ra một SDU với 1 bit start, 7 bit số liệu, 1 bit chẵn lẻ (mức lẻ được để cố định) và 1 bit stop. Lưu ý rằng bit start luôn bằng 0 (space) và bit stop luôn bằng 1 (mark).



bit start 7bit số liệu bit c/lẻ bit stop

Hình 6-2

1.4. Tiêu chuẩn RS-232C

Các ghép nối của PC cho trao đổi nối tiếp đều theo tiêu chuẩn RS-232C của EIA (electronic industries association) hoặc của CCITT ở châu Âu. Chuẩn này quy định ghép nối về cơ khí, điện và lôgic giữa một thiết bị đầu cuối số liệu DTE (data terminal equipment) và thiết bị thông tin số liệu DCE (data communication equipment). Thí dụ DTE là PC còn DCE là MODEM. Có 25 đường với đầu cắm 25 chân D -25 giữa DTE và DCE. Hầu hết việc truyền số liệu là không đồng bộ. Có 11 tín hiệu trong chuẩn RS-232C dùng cho PC, IBM còn quy định thêm đầu cắm 9 chân D -9. Bảng sau chỉ ra các chân tín hiệu và mối quan hệ giữa các đầu cắm 25 chân và 9 chân.

Khác với tín hiệu lôgic TTL, tín hiệu RS -232C là lưỡng cực: mức lôgic 1 (còn gọi là mark) có điện thế âm so với đất trong dải :-3V đến -25V, mức 0 (còn gọi là space) có điện thế dương so với đất và nằm trong dải từ +3 đến +25V. Sở dĩ cần điện thế cao như vậy để có thể truyền được đi xa. Mức logic xác định trong một dải để đề phòng sụt thế trên đường truyền khi điểm cuối ở xa. Thường để phối hợp mức từ máy tính ra giao diện RS232 và ngược lại phải dùng bộ chuyển mức tín hiệu là vi mạch MAX232 hoặc C232.

D-25

D-25	D-9	Tín hiệu	Hướng truyền	Mô tả
1	-	-	-	Protected Ground: nối đất bảo vệ
2	3	TD	DTE→DCE	Transmitted Data: số liệu phát
3	2	RD	DCE→DTE	Received Data: số liệu thu
4	7	RTS	DTE→DCE	Request To Send: DTE yêu cầu truyền số liệu
5	8	CTS	DCE→DTE	Clear To Send: DCE sẵn sàng nhận số liệu
6	6	DSR	DCE→DTE	Data Set Ready: DCE sẵn sàng làm việc
7	5	GND	-	Ground: nối đất (0V)
8	1	DCD	DCE→DTE	Data Carier Detect: DCE phát hiện sóng mang
20	4	DTR	DTE→DCE	Data Terminal Ready: DTE sẵn sàng làm việc
22	9	RI	DCE→DTE	Ring indicator: báo chuông
23	-	DSRD	DCE→DTE	Data Signal Rate Detector: dò t/độ truyền

Chuẩn RS-232C cho phép truyền tín hiệu với tốc độ đến 20 000 baud nhưng nếu cáp truyền đủ ngắn có thể lên đến 115 200 baud. Chiều dài cáp cực đại là 17-20m.

2. Thâm nhập cảng nối tiếp qua DOS và BIOS

Bằng lệnh ngoại trú MODE của DOS có thể đặt các thông số cho cổng nối tiếp RS-232. Ví dụ :

MODE COM2 : 2400,E,8,1

; chọn cổng COM2, tốc độ 2400 baud, chẵn-lẻ chẵn, 8 bit số liệu và 1 bit stop.

Cũng có thể dùng ngắt 21h của DOS để phát hoặc thu số liệu qua cổng nối tiếp bằng 4 hàm như sau:

Hàm 03h : đọc một kí tự từ cổng.
Hàm 04h : phát một kí tự ra cổng.
Hàm 3Fh : đọc một file qua cổng.
Hàm 40h : viết một file qua cổng.

BIOS cho phép thâm nhập khối ghép nối nối tiếp qua ngắt số 14h. Bằng hàm 00h có thể khởi động khối ghép nối, khởi tạo định dạng dữ liệu cũng như tốc độ truyền... Hàm 01h và 02h cho phép phát và thu một kí tự; hàm 03h trả về trạng thái của cổng nối tiếp. Các hàm 04h và 05h cho phép mở rộng các điều kiện khởi động khối ghép nối cũng như cho phép thâm nhập các thanh ghi điều khiển Modem. Các hàm này sau khi được thực hiện đều trả về byte trạng thái trong thanh ghi AH. Một vài hàm cũng cung cấp một byte trạng thái Modem trong thanh ghi AL. Cấu trúc của hai loại byte này như sau:

Byte trạng thái phát

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

D_7: Lỗi quá thời gian (time-out)
1 = có lỗi 0 = không lỗi
D_6: Thanh ghi dịch phát
1 = rỗng 0 = không lỗi
D_5: Thanh ghi đệm phát
1 = rỗng 0 = không lỗi

Byte trạng thái Modem

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

D_7: Phát hiện sóng mang
1 = phát hiện 0 = không
D_6: Chỉ báo tín hiệu chuông
1 = có 0 = không
D_5: Tín hiệu DTR
1 = có 0 = không

D_4 : Ngắt đường truyền

1 = có   0 = không

D_3 : Lỗi khung truyền SDU

1 = có   0 = không

D_2 : Lỗi chẵn lẻ

1 = có   0 = không

D_1 : Lỗi tràn

1 = có   0 = không

D_0 : Số liệu thu

1 = có   0 = không

D_4 : Tín hiệu CTS

1 = có   0 = không

D_3 : Tín hiệu DDC

1 = có   0 = không

D_2 : Tín hiệu delta RI

1 = có   0 = không

D_1 : Tín hiệu delta DTR

1 = có   0 = không

D_0 : Tín hiệu delta CTS

1 = có   0 = không

Thanh ghi DX chứa giá trị tương ứng với các cảng cần truy xuất (00h cho

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

COM_1, 01h cho COM_2, 10h cho COM_3 và 11h cho COM_4). Các thông số định dạng khung truyền SDU được nạp vào thanh ghi AL theo nội dung như sau:

D_7, D_6, D_5 : Tốc độ baud

000 = 110 baud   001 = 150 baud   010 = 300 baud   011 = 600 baud

100 = 1200 baud   001 = 2400 baud   110 = 4800 baud   111 = 9600 baud

D_4-D_3: Bit chẵn lẻ   00 = không có   01 chẵn - lẻ : lẻ   10 = không có
11= chẵn-lẻ : chẵn

D_2 : Số bit stop   0 = 1 bit   1 = 2 bit

D_1-D_0 : Số bit số liệu   10 = 7 bit   11 = 8 bit

Ví dụ, đoạn chương trình sau cho phép khởi động cảng COM_2 với tốc độ 4800 baud, chẵn-lẻ chẵn, 2 bit stop, 7 bit số liệu:

```
MOV  AH, 00h      ; nạp số hàm vào AH
MOV  AL, DEh      ; byte thông số 1101  1110b
MOV  DX, 01h      ; COM2
INT  14h          ; gọi ngắt.
```

Ví dụ, xác định trạng thái của cảng ghép nối COM_1

```
MOV  AH, 03h      ; nạp số hàm vào AH
```

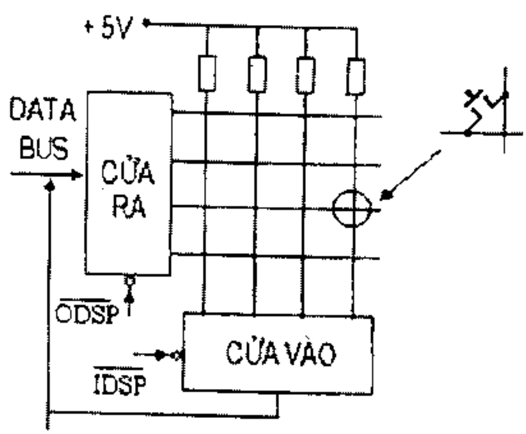
```
        MOV  DX, 00h        ; COM1
        INT  14h            ; gọi ngắt.
```

Ví dụ, đọc một kí tự từ COM_3

```
        MOV  AH, 02h        ; nạp số hàm vào AH
        MOV  DX, 02         ; COM3
        INT  14h            ; gọi ngắt.
```

III. BÀN PHÍM

Bàn phím thường có số lượng phím lớn khoảng 102 phím. Người ta bố trí các phím thành một ma trận chữ nhật, phím được gắn với một công tắc và nằm ở chỗ giao nhau giữa hàng và cột. Mỗi phím được gán cố định một ký tự. Khi nhấn phím thì hàng được nối với cột (Hình 6-3), nếu đó là loại phím cảm biến điện trở (còn hai loại phím khác là cảm biến điện dung hay cảm biến điện từ khi nhấn phím các thông số tương ứng thay đổi).



Hình 6-3

Các hàng được nối với đầu ra của bộ giải mã loại MUX, (lối vào điều khiển của MUX là lối ra của một bộ đếm) sao cho trong một khoảng thời gian nhỏ chỉ có một hàng có giá trị bằng 1, còn các hàng khác có giá trị bằng 0 và giá trị 1 sẽ chạy lần lượt từ hàng này sang hàng khác với tốc độ đủ lớn. Khi một phím được nhấn, do thời gian nhấn phím lâu hơn thời gian chuyển giá trị 1 từ hàng này sang hàng khác, nên tại vị trí phím nhấn cả hàng và cột đều có giá trị 1. Khi đó bộ điều khiển bàn phím sẽ phát ra mã scan (mã quét) của ký tự được nhấn và mã này được viết vào bộ đệm bên trong bàn phím. Mã quét có giá trị lần lượt

từ 01 đến 102 tương ứng với số lượng phím. Việc tạo mã quét có thể được thực hiện bằng phần cứng hay phần mềm. Trong các bàn phím gần đây, vi mạch điều khiển bàn phím 8048 (gồm một CPU, ROM chứa chương trình điều khiển bàn phím, RAM, bộ định thời -timer và hai cổng P1,P2 loại 8b) có nhiệm vụ tính vị trí phím nhấn, tạo mã quét bằng phần mềm và truyền mã này một cách nối tiếp đồng bộ (lai : vừa có bit đồng bộ, vừa có xung nhịp ở chân cắm số 1- xem bảng mô tả chân cắm) tới mạch ghép nối bàn phím (chip vi điều khiển 8042) trong PC. Bảng sau chỉ ra cấu trúc của SDU cho việc truyền số liệu này:

SDU

0										10
STRT	DB_0	DB_1	DB_2	DB_3	DB_4	DB_5	DB_6	DB_7	PAR	STOP

STRT: Bit start (luôn bằng 0)
DB_0 - DB_7: Bit số liệu từ 0 đến 7
PAR: Bit chẵn lẻ (luôn lẻ)
STOP: Bit Stop (luôn bằng 1)

Từ máy AT trở đi, bộ giao diện bàn phím 8042 được chương trình hóa để số liệu có thể được truyền theo hai hướng từ bàn phím và mạch ghép nối, do vậy bàn phím có thể nhận lệnh điều khiển từ PC, ví dụ như đặt tốc độ lặp lại của nhấn bàn phím, để khóa bàn phím...

Chân cắm đầu nối bàn phím

Chân	Tín hiệu
1	Xung nhịp bàn phím
2	Số liệu bàn phím
3	Reset bàn phím hoặc không dùng
4	Đất
5	Vec (+5V)
Bọc kim	Đất bảo vệ

Mỗi một phím nhấn được gán cho một mã quét (scan code) gồm 1 byte. Điều này là đủ cho các bàn phím có tới 102 phím như hiện nay. Nếu một phím

được nhấn, mạch ghép nối bàn phím sẽ phát ra ngắt cứng IRQ_1 tương ứng với INT 09h. Địa chỉ bộ đệm dữ liệu bàn phím là 60h. Chương trình xử lý ngắt sẽ chuyển mã quét thành mã ASCII và đặt mã này vào bộ đệm trong RAM tại địa chỉ đoạn 40, dung lượng 15 từ. Byte cao của từ chứa mã scan, còn byte thấp chứa mã ASCII (nếu là phím ASCII) hay bằng 0 nếu là phím chức năng. Như vậy, các phím chức năng được phân biệt chỉ bởi mã scan. Một số phím có tác dụng đặc biệt (ví dụ có tác dụng lâu dài chỉ sau một lần nhấn), khi nhấn sẽ đặt một cờ lên 1. Cờ là một bit thuộc byte có địa chỉ 0400:0017.

Nếu rời tay nhấn thì bàn phím sẽ phát ra break-code. Mã này giống mã scan nhưng bit 7 được đặt lên 1, do vậy nó tương đương với mã scan cộng với 128 để bộ điều khiển biết được phím đã được nhả, nếu không có tín hiệu nhả phím thì 8042 sẽ tự động gửi mã đã ấn liên tục về vi xử lý sau 0,5s, nhưng thông tin không được lưu trữ, trừ một vài phím liên quan đến cờ. Phần cứng và phần mềm xử lí bàn phím còn phải giải quyết những vấn đề về vật lí sau:

- Nhấn và nhả phím nhưng không được phát hiện.
- Khử nhiễu rung cơ khí và phân biệt một phím được nhấn nhiều lần hay được nhấn chỉ một lần nhưng được giữ trong khoảng thời gian dài (sử dụng RS-FF chỉ thay đổi trạng thái một lần mặc dù nhiều tác động ở một lối vào).

IV. CHUỘT

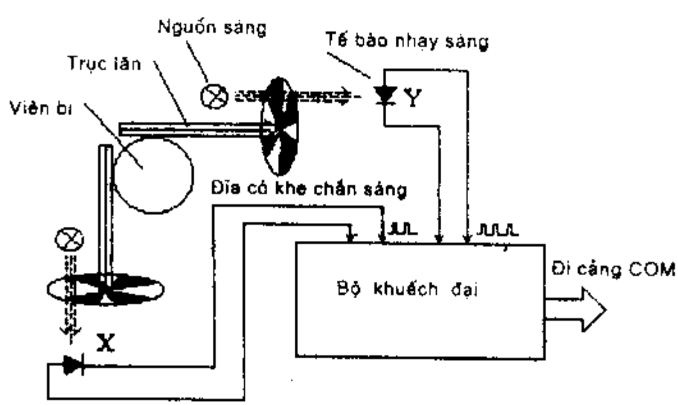
Chuột có nhiều loại, nếu phân loại theo cách ghép nối với máy tính thì có chuột song song do được ghép qua cổng LPT. Chuột nối tiếp (hữu tuyến) ghép qua cổng COM, nối vô tuyến qua cổng hồng ngoại hay nối qua vi điều khiển 8042 như chuột PS/2. Nếu phân loại theo nguyên tắc đo vận tốc chuyển động thì có chuột cơ quét tiếp xúc, chuột cơ quét quang, quả cầu viết (thực ra là chuột cơ lật ngược). Đặc biệt, trong nhóm sau có chuột quang. Chuột quang luôn phải đi kèm với bàn chuột (mouse pad) đặc biệt. Bàn có những đường màu đỏ (trục X) và màu xanh (trục Y) trên một nền phản quang, bên trong chuột bố trí hai hệ thống quét laser khác nhau, một loại bị màu đỏ hấp thụ, loại kia bị màu xanh hấp thụ. Tia không bị hấp thụ sẽ phản xạ trở lại từ đó ta xác định được toạ độ, hướng chuyển động của chuột. Tuy nhiên, chúng ta đi sâu nghiên cứu chuột cơ quét quang là loại thông dụng.

Cấu tạo của chuột (mouse) rất đơn giản, phần trung tâm là một viên bi thép được phủ keo hoặc nhựa và được quay khi dịch chuyển chuột. Chuyển động này được truyền tới hai thanh nhỏ được đặt vuông góc với nhau. Các thanh này sẽ

biến chuyển động của chuột theo hướng X và Y thành sự quay tương ứng của hai đĩa gắn với chúng. Trên hai đĩa có những lỗ nhỏ liên tục đóng và ngắt hai chùm ánh sáng tới các sensor nhạy sáng để tạo ra các xung điện. Vậy số các xung này tỉ lệ với lượng chuyển động của chuột theo các hướng X và Y và số xung trên một giây biểu hiện tốc độ của chuyển động chuột. Để xác định hướng chuyển động của chuột theo trục X cũng như trục Y còn có thể có thêm một cặp điôt phát quang và sensor nhạy quang nằm ở phía đối diện cặp vẽ trên hình. Chuột còn có hai hoặc ba phím bấm để đưa tín hiệu chọn vị trí hiện hành. Hình 6-4 là sơ đồ cấu tạo của chuột

Hầu hết chuột được nối với PC qua các cổng ghép nối nối tiếp, qua đó chuột cũng được cấp nguồn nuôi từ PC. Khi dịch chuyển hoặc nhấn, nhả các phím chuột, nó sẽ phát ra một gói các số liệu tới mạch ghép nối và mạch đó sẽ phát ra một ngắt của cổng COM (IRQ_3=INT0Bh, IRQ_4=INT0Ch lần lượt cho COM_2 và COM_1). Tuy nhiên chương trình điều khiển chuột (mouse driver) đã chiếm các ngắt này và thay đổi địa chỉ chương trình con phục vụ ngắt giao diện nối tiếp. Phần mềm điều khiển chuột còn làm các nhiệm vụ: chuyển ngắt tới mạch ghép nối nối tiếp xác định, đọc gói số liệu và cập nhật các giá trị bên trong liên quan tới trạng thái của phím cũng như vị trí của chuột. Hơn nữa, nó còn cung cấp một ghép nối mềm qua ngắt chuột số 33h để dịch các giá trị bên trong này cũng như làm dịch chuyển con trỏ chuột trên màn hình tương ứng với vị trí của chuột.

Có thể chọn kiểu con trỏ chuột cứng hoặc mềm trong chế độ văn bản hoặc con trỏ chuột đồ họa trong chế độ đồ họa. Các hàm 09h và 0Ah trong ngắt 33h cho phép định nghĩa loại và dạng con trỏ chuột.



Hình 6-4

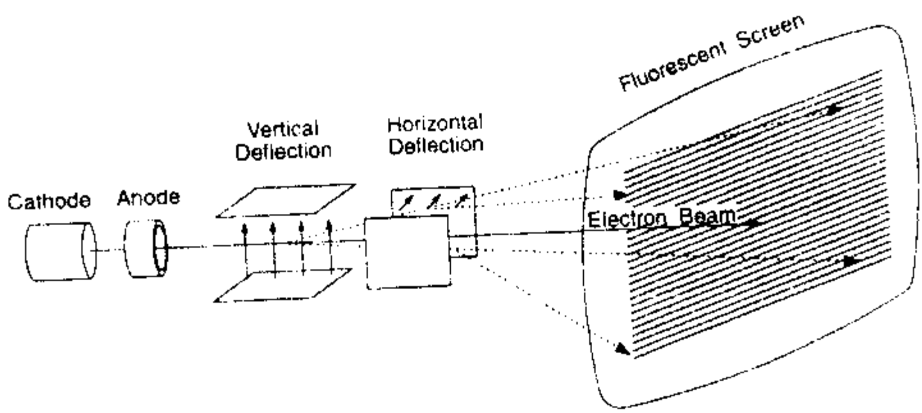
Hàm 09h định nghĩa con trỏ chuột trong chế độ đồ họa. Hiện con trỏ trong chế độ văn bản có phần dễ hơn với hàm 0Ah. Thí dụ, đoạn chương trình sau cho phép hiện con trỏ mềm với màu số 3 và sáng nhấp nháy:

```
MOV     AX, 0Ah        ; chọn hàm
MOV     BX, 00h        ; con trỏ chuột mềm
MOV     CX, 00h        ; xóa kí tự trên màn hình
MOV     DX, 8B02
; BLNK = 1b, BAKx = 000b, INT = 1b, CHRx = 00000010b
INT     33h            ; gọi ngắt.
```

V. MONITOR VÀ BẢN MẠCH GHÉP NỐI MÀN HÌNH

1. Nguyên lí hiện ảnh trên màn hình

Màn hình có nhiều loại như màn hình tia âm cực CRT (cathod ray tube), màn hình tinh thể lỏng LCD (liquid crystal display), màn hình plasma PD (plasma display). Trong phần này chúng ta sẽ đi sâu vào loại màn hình thông dụng nhất CRT.



Hình 6-5

Phương pháp hiện ảnh trên màn hình của monitor máy tính cũng giống như trong máy thu hình thông thường. Hình 6-5 minh họa việc hiện ảnh trên màn hình kiểu ống phóng tia âm cực CRT. Đèn hình thực chất là một đèn điện tử, phía đầu loa rộng dùng để quan sát hình ảnh là anôt gồm một lớp huỳnh quang phủ ở phía trong đầu loa rộng và gần đó là một lưới bằng kim loại. Lưới này được nối với điện áp rất cao (18KV-24KV) để gia tốc các tia điện tử phát ra từ

một súng đặt ở đầu nhỏ bên kia (catôt) của đèn. Tia điện tử gia tốc đập vào chất huỳnh quang sẽ phát sáng và tạo ra một điểm ảnh, màu của ánh sáng tuỳ thuộc vào loại chất huỳnh quang. Độ sáng phụ thuộc vào cường độ tia điện tử. Màn hình một màu thì trên bề mặt chỉ có một loại chất huỳnh quang. Màn hình màu thì lớp huỳnh quang được hình thành từ rất nhiều phần tử của ba loại hoá chất khác nhau, mà khi tia điện tử đập vào sẽ tạo ra ba màu khác nhau là đỏ (red-R), xanh da trời (blue-B), xanh lá cây (green-G). Về nguyên tắc khi trộn ba màu này với tỷ lệ khác nhau sẽ tạo ra tất cả các màu trong tự nhiên. Một điểm ảnh màu vì thế được tạo bởi ba điểm màu, các điểm này thường được bố trí thành hình tam giác và được đặt cách nhau một phút (nhỏ hơn khả năng phân giải của mắt người nên được coi là một điểm). Ba súng điện tử từ phía catôt sẽ bắn vào các điểm màu này theo từng khối ba màu, cường độ của những tia này được thay đổi nhờ điện áp thay đổi có trên lưới điều khiển đặt sát catôt và do vậy tạo ra các điểm ảnh có màu khác nhau. Để tạo ra hình ảnh trên màn hình phải điều khiển tia điện tử (được phát ra từ một catôt bị nung nóng ở đuôi đèn), di chuyển từ trái sang phải tạo ra dòng (quét dòng) và từ trên xuống dưới tạo ra mành (quét mành), nhờ những nam châm điện từ là những cuộn lái tia. Điện áp điều khiển lái tia đưa vào cuộn lái tia có dạng răng cưa và tăng dần một cách tuyến tính. Để nhìn thấy hình ảnh liên tục cần phải có ít nhất 24 mành tức là 24 hình trong một giây, thường thì các đèn hình có khả năng cho 30-60 hình trong một giây. Những tia điện tử gặp đầu mút bên phải cũng như bên dưới của màn hình sẽ bị tắt ngay và được lái rất nhanh theo hướng ngược lại để lại bắt đầu quá trình quét mới. Có hai cách quét:

- Quét xen kẽ (interlaced): Các dòng lẻ được quét trước cho đến hết màn hình theo chiều dọc, gọi là mành lẻ; sau đó các dòng chẵn tạo nên mành chẵn được quét sau. Phương pháp này có ưu điểm là thu hẹp được dải tần số làm việc của thiết bị nhưng có nhược điểm là hình ảnh bị nhấp nháy.

- Quét không xen kẽ (non - interlaced): Các dòng quét được thực hiện tuần tự. Ưu điểm là hình ảnh có thể được điều chỉnh chính xác và ổn định nhưng tất nhiên thiết kế mạch điện sẽ khó hơn vì phải giải quyết vấn đề tăng dải tần làm việc.

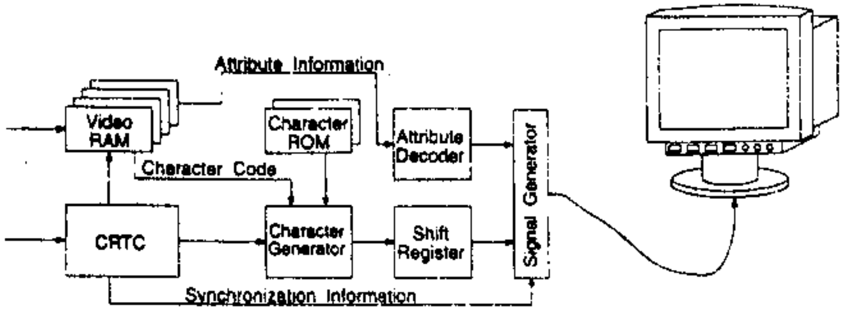
Các monitor dùng màn hình tinh thể lỏng LCD hoặc ống chứa khí được hoạt động theo nguyên lí tương tự như trên nhưng ở đây không có tia điện tử quét, nên thay vì các điểm ảnh riêng biệt là các phần tử phát sáng được định địa chỉ một cách tuần tự. Do vậy, trên các monitor này hình ảnh cũng được phát ra từng dòng một. Quá trình quét ngược cũng không còn nữa vì ở đây đơn giản chỉ việc

thay đổi địa chỉ về phần tử đầu dòng tiếp theo.

Thường màn hình máy tính phải có các mạch dao động tạo xung quét dòng và mành để điều khiển chuyển động của tia điện tử và mạch khuếch đại tín hiệu hình để điều khiển cường độ sáng của tia điện tử. Các tín hiệu này phải được đồng bộ với nhau để một điểm ảnh (pixel) của hình phải được hiện ra tại cùng một vị trí trên màn trong mỗi mành.

2. Bản mạch ghép nối màn hình

Để hiện các hình ảnh kí tự hoặc hình vẽ trên màn hình, PC phải thông qua mạch ghép nối màn hình (graphic adapter). Bản mạch này thường được cắm trên khe cắm mở rộng của PC. Sơ đồ khối của bản mạch cho màn hình ký tự trong hình 6-6.



Hình 6-6

Phần trung tâm là chip điều khiển ống hình CRTC (cathode ray tube controller). CPU thâm nhập RAM video qua mạch ghép nối bus để viết thông tin xác định ký tự hoặc hình vẽ cần hiển thị. CRTC liên tục phát ra các địa chỉ để RAM video đọc các kí tự trong đó và truyền chúng tới máy phát kí tự (character generator). Trong chế độ văn bản (text mode), các kí tự được xác định bởi mã ASCII, trong đó có cả các thông tin về thuộc tính của kí tự, thí dụ kí tự được hiện theo cách nhấp nháy hay đảo màu đen trắng... Mỗi kí tự được biểu diễn bởi một từ hai byte trong RAM video. Byte thấp chứa mã kí tự, byte cao chứa thuộc tính. Cấu trúc của một từ nhớ video như sau:

15							7								0
BLNK	BAK_2	BAK_1	BAK_0	INT	FOR_2	FOR_1	FOR_0	CHR_7	CHR_6	CHR_5	CHR_4	CHR_3	CHR_2	CHR_1	CHR_0

BLNK : (nhấp nháy) 1 = bật, 0 = tắt

BAK_2..BAK_0 : Màu nền (từ bảng màu hiện tại)

INT : Cường độ sáng : 1 = cao, 0 = bình thường

FOR_2.. FOR_0 : Màu nền trước (từ bảng màu hiện tại)

CHR_7.. CHR_0 : Mã kí tự

ROM kí tự (character ROM) lưu trữ các hình mẫu điểm ảnh của các kí tự tương ứng để máy phát ký tự biến đổi các mã kí tự đó thành một chuỗi các bit điểm ảnh (pixel bits) và chuyển chúng tới thanh ghi dịch (shift register). Thường mẫu ký tự là một ma trận 9x14 điểm ảnh. Máy phát tín hiệu sẽ sử dụng các bit điểm ảnh này cùng với các thông tin thuộc tính từ RAM video và các tín hiệu đồng bộ từ CRTC để phát ra các tín hiệu cần thiết cho monitor.

Trong chế độ đồ họa (graphics mode), thông tin trong RAM video được sử dụng trực tiếp cho việc phát ra các kí tự, hình ảnh. Lúc này các thông tin về thuộc tính cũng không cần nữa. Chỉ từ các giá trị bit trong thanh ghi dịch, máy phát tín hiệu sẽ phát các tín hiệu về độ sáng và màu cho monitor. Các mẫu ký tự có thể lấy từ một file chứa các loại phông chữ và nạp vào RAM video. Màn hình đồ hoạ đen trắng (một màu) thường có 640x400 pixel (độ phân giải), mỗi điểm ảnh được điều khiển độ sáng bởi một bit, ví dụ bit 0 cho độ sáng nền và bit 1 cho độ sáng của điểm ảnh. Do vậy RAM video có dung lượng: 640x400b/8=32000B, cỡ 32KB. Tuy nhiên cần có hai bộ nhớ như vậy và gọi là hai trang màn hình. Một trang để nhận tin trong khi trang kia chuyển tin lên màn hình và ngược lại thì hình ảnh mới liên tục, do vậy bộ nhớ tối thiểu gồm hai trang màn hình có kích thước S=64KB.

Hiện nay màn hình màu VGA là mạnh và thông dụng nhất, chúng ta sẽ đi sâu nghiên cứu bản mạch cho nó. Bản mạch thông dụng VGA màu (video graphics adapter), có độ phân giải tốt nhất 1024x768 điểm ảnh, mỗi điểm ảnh dùng18 bit trong đó dùng 6 bit cho một màu, vì thế số màu tối đa là 2^{18} = 262144 màu, hay 256K màu. Có thể tính dung lượng bộ nhớ màn hình như sau : mỗi điểm ảnh cần 18b, nếu dùng màn hình có độ phân giải 1024x768 thì cần bộ nhớ kích thước: 1024x768x18 =14155776b =1769472B. Tuy nhiên, như đã nói ở trên cần phải có hai bộ nhớ như vậy, do vậy bộ nhớ tối thiểu gồm hai trang màn hình phải có dung lượng S cỡ 4MB. Trong trường hợp phát những hình ảnh tốc độ cao, độ phân giải lớn hơn dung lượng S có thể lên tới 16 MB. Cần lưu ý rằng, bộ nhớ màn hình chỉ được dành địa chỉ từ A0000h đến C0000h

trong bản đồ bộ nhớ chính (128KB), do vậy bộ nhớ màn hình được hoạt động theo chế độ chuyển mạch mềm từng khối 128 KB một. Với màn hình một màu chỉ cần một dây để đưa một bit vào mạch tạo tín hiệu điều khiển cường độ tia điện tử bên màn hình (monitor). Nhưng bản mạch VGA không thể phát tín hiệu số tới monitor vì phải cần đến 18 đường tín hiệu số. Để tránh điều đó, bản mạch VGA phát ra tín hiệu tương tự để điều khiển màn hình. Ba DAC (bộ biến đổi số-tương tự/digital-analog convertor) 6 bit sẽ nhận 6 bit điều khiển (ứng với 3 màu) ở lối vào, ở lối ra sẽ cho ra tín hiệu tương tự mà biên độ của nó tỷ lệ thuận với giá trị thập phân của các bit tác động ở lối vào. Do vậy chỉ cần 3 dây truyền các tín hiệu màu tương ứng sang phía màn hình là đủ.

Trong thực tế không phải lúc nào bộ nhớ màn hình cũng đủ lớn để hiển thị tất cả các màu với độ phân giải mong muốn, trong trường hợp đó có khi chỉ cần hiển thị 16 màu, 256 màu hay 16 bit màu (high color 16 bit) là đủ, tức là giảm số bit cần thiết cho việc lưu giữ thông tin của điểm ảnh màu. Muốn vậy người ta sử dụng việc mã hoá thông tin theo bảng màu (palette). Trong một thời điểm, người ta chỉ sử dụng một số màu trong bảng màu này. Ví dụ như là chỉ sử dụng 256 màu trong 256K màu, khi đó một điểm ảnh màu chỉ cần 8 bit; hay chỉ sử dụng 16 màu trong bảng màu 256 màu thì một điểm màu chỉ cần 4 bit. Một bảng màu có dung lượng tối đa cho phép hiển thị điểm ảnh ở các chế độ màu khác nhau tuỳ theo dung lượng bộ nhớ màn hình đang có. Sự phong phú của gam màu vẫn được đảm bảo nhờ việc chọn một nhóm màu bất kỳ trong bảng màu, tuy nhiên tại một thời điểm ta chỉ được sử dụng duy nhất một nhóm màu đã chọn khi bộ nhớ màn hình nhỏ.

Trên bản mạch VGA người ta sử dụng 256 thanh ghi màu 18b chia thành 3 nhóm, mỗi nhóm 6b nối vào 3 DAC tương ứng kể trên. Các thanh ghi này chia thành 16 nhóm màu (từ nhóm 0 đến 15) mỗi nhóm gồm 16 thanh ghi màu. Mỗi thanh ghi màu chứa thông tin để hiển thị một màu của điểm ảnh và 256 thanh ghi chứa thông tin của 256 màu, có thể thấy rằng mới chỉ sử dụng có 8b trong mỗi thanh ghi. Giả sử ta chỉ có đủ bộ nhớ cho chế độ 16 màu, khi đó một điểm ảnh chỉ cần 4b và 4b này trong bộ nhớ màn hình sẽ được dùng để chọn 1 trong 16 thanh ghi màu trong một nhóm nối vào DAC. Bốn bit thấp của một thanh ghi chọn màu (không nằm trong số các thanh ghi trên) được dùng để chọn 1 trong 16 nhóm màu. Thay đổi nội dung của thanh ghi chọn màu ta sẽ được các nhóm màu khác nhau. Trong chế độ 256 màu, một điểm ảnh cần 8b, 8b này sẽ

chọn ra 1 trong số 256 thanh ghi màu. Để chọn các nhóm 256 màu khác nhau phải can thiệp vào bit chưa sử dụng của các thanh ghi màu. Trong chế độ sử dụng dung lượng lớn hơn của bảng màu phải nạp toàn bộ số bit cho một điểm ảnh màu vào nội dung của thanh ghi màu. Do tổ chức sẵn 256 thanh ghi màu trong các chế độ 16 hay 256 màu ta dễ dàng chọn màu một cách hài hoà trong một nhóm, giữa các nhóm sao cho chất lượng ảnh màu tốt nhất, đồng thời tốc độ chuyển màu và nhóm màu sẽ rất nhanh. Một phần mềm (driver) sẽ trợ giúp việc chọn chế độ màu cũng như độ phân giải thích hợp. Các thanh ghi màu thường được tích hợp với các DAC trong một chip gọi là RAMDAC.

Ngoài RAM video và RAMDAC trên bản mạch VGA còn có hai chip quan trọng khác là chip điều khiển bản mạch và ROM của riêng bản mạch. Vi mạch điều khiển dùng để điều phối hợp hoạt động trên bản mạch, trong đó quan trọng nhất là thực hiện chế độ DMA để làm tươi hình ảnh trên màn hình. Như trên đã đề cập, một hình ảnh phải được đưa ra cỡ 30-60 lần trong một giây, người ta không muốn CPU làm việc này và do vậy chip điều khiển sẽ thực hiện điều đó trong chế độ DMA. Chip ROM có nhiệm vụ bổ sung cho ROMBIOS trong việc chọn chế độ, độ phân giải cho màn hình, các trang của RAM video, tạo các chuyển mạch mềm (chặn ngắt 10h...), nâng cao khả năng xử lý đồ họa. BIOS của VGA được tăng cường thêm hàm 1C cho phép sao lưu các trạng thái của video rất quan trọng trong chế độ đa nhiệm, vì chế độ này thường xuyên biến đổi các chế độ video.

Bảng sau cho các thông số của bản mạch VGA.

	Chế độ văn bản	**Chế độ đồ họa**
Đoạn video	B000h	B000h
Kích thước RAM video	256 kbyte	256 kbyte
Các trang màn hình	1..8	1..8
Bộ điều khiển video	VGA-CRTC	VGA-CRTC
Địa chỉ cảng của chip điều khiển video 6845	3B0h-3DFh	3B0h-3DFh
Ma trận kí tự	9x16	9x16
Kích thước kí tự hiệu dụng	7x9	7x9
Độ phân giải (pixels)	640x480	640x480

Các màu	256	256
Tín hiệu điều khiển monitor	analog	analog
Tần số quét ngang	31,5 kHz	31,5 kHz
Tần số quét dọc	50-70 Hz	50-70 Hz
Bề rộng dải tần video	28 MHz	28 MHz
BIOS của riêng bản mạch	có	có

3. Thâm nhập màn hình qua DOS và BIOS

3.1. Thâm nhập qua DOS

Qua các hàm của INT 21h có thể hiện các ký tự trên màn hình nhưng không can thiệp được vào màu:

- Hàm 02 h ra màn hình
- Hàm 06 h ra một ký tự
- Hàm 09 h ra một chuỗi
- Hàm 40 h viết file/thiết bị

Từ DOS 4.00 trở đi có thể dùng lệnh MODE để điều chỉnh số cột văn bản từ 40 đến 80 hoặc số dòng từ 25 đến 50.

Các lệnh COPY, TYPE và PRINT trong mức lệnh COMMAND.COM cho phép hiện text trên màn hình. DOS gộp chung bàn phím và monitor thành một thiết bị mang tên CON (console). Viết tới CON tức là luôn truyền số liệu tới monitor còn đọc CON là nhận các ký tự từ bàn phím. Thí dụ, để hiện nội dung của file output.txt lên màn hình của monitor sẽ có ba cách :

- COPY output.txt CON
- TYPE output.txt > CON
- PRINT output.txt /D:con

3.2. Thâm nhập qua BIOS

Bằng INT 10h, BIOS thâm nhập monitor với nhiều chức năng hơn DOS, thí dụ đặt chế độ hiện hình, quản lý tự động các trang, phân biệt các điểm trên màn hình nhờ các tọa độ... BIOS có sẵn những hàm dùng cho thâm nhập các loại bản mạch MDA và CGA. BIOS của riêng VGA có những hàm mở rộng tương ứng trong khi vẫn giữ nguyên định dạng gọi. Một trong những hàm quan trọng nhất của INT 10h là hàm 00h dùng để đặt chế độ hiện hình. Thí dụ, khởi tạo kiểu 6 với độ phân giải 640x200 trên CGA.

```
MOV AH, 00h   ; hàm 00h
MOV AL, 06h   ; chế độ 6
INT      10h  ; gọi ngắt
```

Các bản mạch VGA (kể cả EGA) có riêng BIOS của chúng. Trong quá trình khởi động PC, nó sẽ chặn INT 10h lại và chạy thường trình BIOS của riêng bản mạch. Thường trình cũ (của BIOS trên bản mạch chính) được thay địa chỉ tới INT 42h. Tất cả các lệnh gọi INT 10h sẽ được BIOS của EGA/VGA thay địa chỉ tới INT 42h nếu bản mạch EGA/VGA đang chạy các kiểu hiện tương thích với MDA hoặc CGA. Có các kiểu hoạt động từ 0 đến 7.

BIOS của EGA/VGA dùng vùng 40:84h tới 40:88h để lưu trữ số liệu BIOS và các thông số của EGA/VGA. Nó có các hàm mới với các hàm phụ sau:

Hàm 10h thâm nhập các thanh ghi màu và bảng màu

Hàm 11h cài đặt các bảng định nghĩa ký tự mới

Hàm 12h đặt cấu hình hệ con video

Hàm 1Bh thông tin về trạng thái và chức năng của BIOS video (chỉ có ở VGA)

Hàm 2Ch trạng thái save/restore của video (chỉ có ở VGA).

Sau đây là một chức năng của các hàm và thí dụ về sử dụng chúng.

- Hàm 10h, hàm phụ 03h - xoá/ đặt thuộc tính.

Ví dụ, xoá thuộc tính nhấp nháy

```
MOVAH, 10h   ; dùng hàm 10h
MOV AL, 03h  ; dùng hàm phụ 3
MOV BL,00h   ; xoá thuộc tính nhấp nháy
INT 10h      ; gọi ngắt
```

- Hàm 11h - ghép nối với máy phát ký tự

Ví dụ, nhập bảng định nghĩa ký tự 8x14 không cần chương trình CRTC

```
MOV AH, 11h  ; dùng hàm 11h
MOV AL, 01h  ; nạp bảng ký tự từ ROM BIOS vào RAM máy phát ký tự
MOV BL, 03h  ; gán số 3 cho bảng
INT     10h  ; gọi ngắt
```

- Hàm 12h, hàm phụ 20h - chọn thường trình in màn hình. Dùng hàm phụ này có thể thay thế thường trình chuẩn cho INT 05h bằng thường trình có thể dùng cho các độ phân giải mới của EGA/VGA.

Ví dụ, cho phép thường trình mới in màn hình

MOV AH, 12h ; dùng hàm 12h

MOV BL, 20h ; dùng hàm phụ 20h.

n PRINT hoặc SHIFT + PRINT để gọi thường trình in đã được lắp đặt.

4. Tương quan địa chỉ

Bản mạch VGA có nhiều chế độ đồ họa 4 đến 6 và 13 đến 19. Xét ví dụ với kiểu 17 đến 19. Kiểu 17 với 80 byte trên một dòng (640 điểm ảnh/8 điểm ảnh trên một byte). Mỗi trang màn hình gồm cỡ 40 Kbyte. Địa chỉ của byte trong bộ nhớ tương ứng với điểm ảnh ở dòng i, cột j (i =0 - 479, j = 0 - 639) được tính như sau

address (i,j) = A0000h + 50h x j + INT (i/8).

Kiểu 18, 4 bit của điểm ảnh được phân trong 4 lớp nhớ như ở EGA. Trong kiểu VGA phân giải cao với 16 màu khác nhau, 80 byte trên một dòng (640 điểm ảnh/8 điểm ảnh trên một byte) mỗi trang màn hình gồm 40Kbyte (A0000h byte); địa chỉ của mỗi byte ở dòng i, cột j (i = 0 - 479, j = 0 - 639) bằng:

address (i,j) = A0000h + 50h x j + INT (i/8)

Kiểu 19 với 256 màu cho một điểm ảnh thì RAM video lại được tổ chức rất đơn giản như một dãy tuyến tính, trong đó 1 byte tương ứng với 1 điểm ảnh. Giá trị của byte phân định màu của điểm ảnh. Kiểu này đòi hỏi 320 byte (140h) trên một dòng (320 điểm ảnh/1 điểm ảnh trên một byte). Một trang màn hình gồm 64 kbyte (10000h) nhưng chỉ có 64000 byte được sử dụng. Địa chỉ của điểm ảnh trong dòng i, cột j (i = 0-199, j = 0-319) là :

address (i, j) = A0000h + 140h x j + i

VI. GHÉP NỐI USB (universal serial bus)

Giao diện tuần tự đa năng USB trong PC nhằm đáp ứng nhu cầu một giao diện đơn giản, giá thành thấp, linh hoạt, chuẩn hoá và dễ sử dụng. USB cho phép ghép nối giữa máy vi tính điện thoại, camera và tất cả các thiết bị chuẩn của PC, nó mở ra khả năng kết thúc tình trạng chắp vá, không đồng nhất của các giao diện ngoại vi trong máy tính cá nhân. USB là giao thức truyền dữ liệu tuần tự giữa máy vi tính (chủ USB) với các thiết bị ngoại vi. Nguyên tắc kết nối này hoàn toàn tương tự cách trao đổi thông tin trong mạng máy tính. Dữ liệu được truyền trên USB theo hai chế độ: 12 Mbs (full speed mode) và 1,5 Mbs (low speed mode). Thiết bị ngoại vi làm việc ở vị trí tớ được nối trực tiếp với máy tính chủ hay gián tiếp qua hub. HUB là bộ tiếp nối sử dụng cấu trúc hình sao

và mỗi hub đều nằm ở trung tâm hình sao đó. Các hub thứ cấp được nối vào hub lớp trên như là một thiết bị ngoại vi. Một ưu điểm quan trọng của USB là hot plug and play nghĩa là có thể cắm hoặc tháo thiết bị mà không cần tắt máy chủ hay cài đặt lại hệ thống. Hệ điều hành sẽ tự động làm việc này. Do sử dụng 7 đường địa chỉ, chủ USB có thể quản lý tối đa 127 thiết bị ngoại vi.

Cáp nối USB gồm 4 dây. Hai dây (D+ và D-) được sử dụng để truyền dữ liệu theo phương pháp vi sai. Hai dây còn lại dùng để cấp nguồn cho thiết bị ngoài và dây đất. Cáp rẻ tiền không bọc kim dùng để truyền tần số thấp, cáp bọc kim cho tần số cao. Phích cắm chia làm hai loại: loại A nối vào máy tính, loại B nối vào thiết bị ngoại vi, hai phích này không đổi chỗ cho nhau được để tránh nhầm lẫn. Thiết bị nhanh có điện trở nối đất cho đầu D+, còn thiết bị chậm cần điện trở nối nguồn cho đầu D-. Căn cứ vào đó máy chủ phát hiện được việc cắm hay rút một thiết bị khỏi mạng USB.

Thiết bị USB có chức năng tớ trong mạng USB và có hai loại : hub gồm một bộ điều khiển hub và một bộ lặp lại (repeat)nhằm chuyển một ổ USB thành nhiều ổ cắm. Thiết bị chức năng là các thiết bị ngoại vi chuẩn USB. Nói chung hai loại thiết bị này gồm : SIE (serial interface engine) đó là một mạch tích hợp, ví dụ như bộ vi điều khiển USB8x930 chịu trách nhiệm nhận và gửi dữ liệu theo giao thức USB. Thành phần thứ hai là một tổ hợp phần cứng và phần sụn chịu trách nhiệm truyền dữ liệu giữa SIE và điểm cuối (thanh ghi dữ liệu) của thiết bị qua những giao diện thích hợp (đường ống). Giao diện này là phần mềm quản lý điểm cuối của chủ USB được gọi từ phần mềm điều hành USB. Thành phần thứ hai này có trong vi mạch USB8x930, vi mạch này có khả năng làm việc với cả hai phía: phía giao diện USB nó sẽ tiếp nhận tín hiệu khởi động từ máy chủ và trả lời nó; phía thiết bị nó có khả năng quản lý thiết bị ngoại vi như các mạch giao diện thông thường bằng các phần sụn có bên trong. Ví dụ nếu thiết bị ngoại vi là một bàn phím thì 8x930 đóng vai trò của chip điều khiển bàn phím cổ điển 8048. Thành phần cuối cùng của thiết bị USB chính là phần chức năng của thiết bị ngoại vi.

Giao thức USB gồm 3 gói chính: gói khung (tocken paket-TP), gói dữ liệu (data paket-DP) và gói bắt tay (handshack paket-HP). Ngoài ra còn một gói khởi đầu khung (start of frame-SOF). Máy chủ USB sẽ duy trì hoạt động bằng cách truyền gói khởi đầu khung SOF trong chu kỳ 1ms. Dải tần của bus sẽ được chia bởi các thiết bị ngoại vi với thời gian 1ms này. Mỗi lần truyền tin cần đến 3 gói

chính nói trên. Một cuộc truyền đang duy trì sẽ được bắt đầu khi máy chủ gửi gói TP gồm dịa chỉ thiết bị ADDR, số hiệu điểm cuối ENDP, hướng của cuộc truyền và dạng ống. Thiết bị có địa chỉ tương ứng sẽ tự chọn bằng cách giải mã địa chỉ của nó từ gói khung. Trường ghi hướng cuộc truyền trong gói khung sẽ xác định tớ hay chủ gửi dữ liệu là gói DP. Sau khi nhận xong dữ liệu bên nhận sẽ gửi gói HP, gói này có thể có 3 ý nghĩa tuỳ tin bên trong: chấp nhận dữ liệu, khong chấp nhận hoặc thông báo tắc STALL.

Gói khung:

8b	7b	4b	5b
PID	ADDR	ENDP	CRC

Gói dữ liệu:

8b	0-1013 byte	16b
PID	DATA	CRC

Gói bắt tay:

8b
PID

Gói khởi đầu khung :

8b	11b	5b
PID	Số khung	CRC

PID (paket identification) : loại gói; ADDR (address) : địa chỉ; ENP (end-point) : điểm cuối; CRC (cyclic redudancy code) : mã kiểm tra vòng dư; DATA : dữ liệu.

Bài tập chương 6

1. Vẽ khung tin truyền nối tiếp không đồng bộ của ký tự A, b, G.

2. Xác định dung lượng bộ nhớ màn hình tối thiểu cho card VGA màu, độ phân giải 640x480

3. Ưu nhược điểm của truyền dữ liệu song song và nối tiếp.

TÀI LIỆU THAM KHẢO

1. ***Cấu trúc máy tính cơ bản*** - Tổng hợp và biên dịch VN - Guide - Nxb Thống kê, 2001.

2. ***Cấu trúc máy vi tính và thiết bị ngoại vi*** - Nguyễn Nam Trung - Nxb Khoa học Kỹ thuật, 2000.

3. ***Lập trình hợp ngữ và máy vi tính IBM-PC*** - Ythayu và Charles Marut

Biên dịch : Quách Tuấn Ngọc, Đỗ Tiến Dũng, Nguyễn Quang Khải - Nxb Giáo dục, 1998.

4. ***Kiến trúc máy tính*** - Nguyễn Đình Việt - Nxb Giáo dục, 2000.

5. ***Cấu trúc máy vi tính*** - Trần Quang Vinh - Nxb Giáo dục, 2000.

MỤC LỤC

BỘ GIÁO TRÌNH XUẤT BẢN NĂM 2005
KHỐI TRƯỜNG TRUNG HỌC KINH TẾ KỸ THUẬT TIN HỌC

1. THUẬT TOÁN VÀ KỸ THUẬT LẬP TRÌNH PASCAL
2. ĐÁNH MÁY VI TÍNH
3. SOẠN THẢO VÀ ĐÁNH MÁY VĂN BẢN
4. NGHIỆP VỤ THƯ KÝ
5. KẾ TOÁN MÁY
6. MARKETING
7. NGÔN NGỮ LẬP TRÌNH C
8. LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG VỚI (C++)
9. BẢNG TÍNH ĐIỆN TỬ (EXCEL)
10. VISUAL BASIC
11. CẤU TRÚC MÁY TÍNH
12. GIAO TIẾP
13. ACCESS
14. MẠNG MÁY TÍNH
15. THIẾT KẾ WEB
16. BẢO TRÌ PC
17. HỆ ĐIỀU HÀNH
18. CƠ SỞ DỮ LIỆU
19. PHÂN TÍCH THIẾT KẾ HỆ THỐNG
20. KỸ THUẬT SỐ
21. PHOTOSHOP
22. CAD/CAM

10154876
8935075903753
Giá: 23.000đ